Universidade de Aveiro
2011
Departamento de Electrónica, Telecomunicações e informática

Tomasz Wojciech Jacel

Técnicas para Garantir Segurança em Comunicações Veiculares

Implementation of a Honeypot for Vehicular Communications

Universidade de Aveiro Departamento de Electrónica, Telecomunicações
2011 e informática

Tomasz Wojciech Jacel

Técnicas para Garantir Segurança em Comunicações Veiculares

Implementation of a Honeypot for Vehicular Communications

Implementacja pułapki typu Honeypot dla bezprzewodowej sieci komunikacji międzypojazdowej

Dissertação apresentada à Universidade de Aveiro para cumprimento dos requisitos necessários à obtenção do grau de Mestre em Engenharia Electrónica e Telecomunicações, realizada sob a orientação científica do Doutor José Alberto Gouveia Fonseca, Professor asociado do Departamento de Electrónica ,Telecomunicações e Informática da Universidade de Aveiro e co-orientação científica do Doutor Joaquim José de Castro Ferreira, Professor Adjunto da Escola Superior de Tecnologia e Gestão de Águeda da Universidade de Aveiro.

Universidade de Aveiro Departamento de Electrónica, Telecomunicações
2011 e informática

Tomasz Wojciech Jacel

Técnicas para Garantir Segurança em Comunicações Veiculares

Implementation of a Honeypot for Vehicular Communications

Implementacja pułapki typu Honeypot dla bezprzewodowej sieci komunikacji międzypojazdowej

Dissertation submitted to the University of Aveiro as part of its Master's in Electronics and Telecommunications Engineering Degree. The work was carried out under the scientific supervision of Professor José Alberto Gouveia Fonseca of the Department of Electronics, Telecommunications and Informatics of the University of Aveiro, and Adjunct Professor Joaquim José de Castro Ferreira of the Superior School of Technology and Management of Águeda from the University of Aveiro.

Mr. Tomasz Wojciech Jacel is a registered student of Technical University of Lodz, Lodz, Poland and carried out his work at University of Aveiro under a joint Campus Europae program agreement. The work was followed by Ph.D Marcin Janicki at Technical University of Lodz as the local Campus Europae Coordinator.

o júri

Presidente

Alexandre Manuel Moutela Nunes da Mota,
Professor Associado do Departamento de Electrónica, Telecomunicações e Informática da Universidade de Aveiro
Francisco Manuel Madureira Vasques de Carvalho,
Professor Associado do Departamento de Engenharia Mecânica, Faculdade de Engenharia da Universidade do Porto
José Alberto Gouveia Fonseca,
Professor Associado, Departamento de Electrónica, Telecomunicações e Informática da Universidade de Aveiro
Joaquim José de Castro Ferreira,
Professor Adjunto da Escola Superior de Tecnologia e Gestão de Águeda da Universidade de Aveiro
Marcin Janicki,
Ph.D of Department of Electrical, Electronic, Computer and Control Engineering of Technical University of Lodz

agradecimentos

I would like to thank my supervisors for guidance over my dissertation. Without them, my stay in Portugal would not be possible.

palavras-chave

Cominicações veículares, segurança, detecção e monitorização de intrusões, pote-de-mel (*honeypot*)

resumo

Esta dissertação descreve um estudo de viabilidade para a implementação de um software do tipo pote-de-mel (honeypot) para comunicações veículares Ad-Hoc sem fios baseadas no protocolo WAVE (Wireless Access in Vehicular Environment).

Um honeypot é uma ferramenta desenhada para simular falsas redes de computadores, monitorá-los, e capturar todos os eventuais comportamentos maliciosos tais como ataques e tentativas de intrusão.

O estudo da solução proposta começa com uma pesquisa de trabalho relacionado e com o estudo dos fundamentos e protocolos das comunicações veículares sem fios, nomeadamente os protocolos IEEE 802.11p e IEEE 1609.2.

De seguida é feito um levantamento dos principais problemas de segurança no âmbito das comunicações veículares sem fios e procede-se a uma descrição detalhada da tecnologia de *honeypots* e é escolhida uma ferramente que irá ser alvo de particular atenção ao longo desta dissertação, o HONEYD.

Finalmente, e dado que esta dissertação tem um caracter iminententemente teórico, são descritas as modificações que serão necessárias para adaptar o HONEYD para comunicações veículares sem fios. Isto para o caso de comunicações veículo a veículo, onde é descrita a integração do HONEYD na unidade de bordo (OBU) e para o caso de ce comunicações veículo a infraestrutura de beira de estrada, onde é proposta uma solução para integração do HONEYD na *road-side-unit* (RSU).

keywords

Vehicular communications, security, intrusion detection and monitoring, honeypot

abstract

This dissertation is an attempt to implement the honeypot software into highly dynamic Vehicular Ad-hoc Network (VANET).This ad-hoc network is based on wireless communication between nodes according to the - WAVE (Wireless Access in Vehicular Environment) protocol. A honeypot is a tool designed to simulate fake local computer networks, monitor them, and capture all malicious behavior aimed towards them. This dissertation is in the scope of Intelligent Transportation Systems (ITS) and it provides some contributions to development of security system and hence, road safety.

Honeypot solution implemented in VANET would help improve security in the network by attracting, catching and analyzing all malicious attempts to break the security system.

The study of proposed solution begins with research and introduction to the main principals of vehicular communication. It is accompanied with system and wireless communication technology description. Presentation of main security issues is also provided.

Honeypot software is also presented by deep in-sight look into its types, functionality, architecture, advantages and disadvantages. Via the research the one type of recent available honeypot is chosen and then deeply scrutinized on the basis of implementation into Vehicular Ad-hoc Network.

Finally, since this dissertation has theoretical character, to-be changes that should be carried out to implement fully the propose solution are provided.

As this work is mainly focused on tailoring and proposing necessary changes to the TCP/IP honeypot software to meet the requirements of WAVE, the hardware tests in real environment as well as creating source code will not be done and are out of scope of this dissertation. Future work should be based on programming necessary modules and putting them into life.

słowa kluczowe

Komunikacja pomiędzy pojazdami, bezpieczeństwo, wykrywanie włamań do sieci, honeypot

streszczenie

Poniższa praca magisterska jest próbą przystosowania programu typu honeypot do działania w mobilnych sieciach ad-hoc - VANET (Vehicular Ad-hoc Network). Sieć ta oparta jest na bezprzewodowej komunikacji pomiędzy pojazdami zgodnie ze standardem WAVE (Wireless Access in Vehicular Environment). Honeypot jest narzędziem służącym do symulowania topologii sieci komputerowej, monitorowania jej i wychwytywania wszelakich prób włamań do niej. Temat tej pracy magisterskiej mieści się w obszarze działalności stowarzyszenia ITS (Intelligent Transportation Systems). Będzie ona miała wpływ na polepszenie bezpieczeństwa w sieciach VANET i co za tym idzie bezpieczeństwa na drogach.

Program honeypot wdrożony w sieciach VANET może w aktywny sposób przyczynić się do poprawienia bezpieczeństwa w sieci, poprzez przyciąganie, wychwytywanie i analizowanie wszelakich prób włamań.

Praca ta zaczyna się przeglądem głównych zagadnień dotyczących bezprzewodowej komunikacji pomiędzy pojazdami w sieciach VANET. Szczególny nacisk jest kładziony na bezpieczeństwo w tych sieciach.

Zaprezentowana jest również idea programu honeypot zarówno jak i jego rodzaje, funkcjonalność, architektura oraz wady i zalety. Poprzez analizę dostępnych programów służących jako honeypot, został wybrany jeden konkretny - honeyd i poddany dokładnej analizie pod kątem implementacji w sieciach VANET.

Jako że ta praca magisterska ma charakter teoretyczny, jej wynikiem jest propozycja funkcjonalności i architektury urządzenia działającego jako honeypot w sieciach VANET. Zaproponowane są również zmiany którym powinien ulec software aby zapewnić pełną komunikację z nowym środowiskiem. Stworzenie kodu źródłowego odpowiadającego tym zmianom oraz testy na sprzęcie są pracą na przyszłość.

Contents

Index of figures

Index of tables

Chapter 1

Introduction

Wireless Access In Vehicular Environment (WAVE) is a recent, innovative technology on the basis of wireless communication. It is designed to provide communication between nodes in Vehicular Ad-hoc Network (VANET). VANET is a network created by vehicles on the roads and devices installed along them. This new type of ad-hoc network was introduced mainly to meet demands of Intelligent Transportation System (ITS) which scope is to increase safety on the roads and to decrease the air pollution by introducing into vehicular traffic modern technologies.

VANET being mainly wireless environment is highly prone to many types of attacks or dishonest behaviors. One of the biggest challenges of WAVE is to provide security of communication. This is achieved so far by many solutions: keying material, certificates distribution or encryption. However, it has been proved that in many situations it is possible to deceive the security system. Intruders are equipped with more and more powerful tools to break the security mechanism. VANET has to have also powerful own countermeasures to fight with them because the reliability of communication via air link has an utmost significance.

The common solution known from the computer networks which gives the security system advantage over the intruder is the honeypot. It is a software used to deceive intruders as well as to reveal theirs tools or hacking methods what is immensely helpful in immunizing the security system.

The current work is based on analysis of the honeypot solution and verification whether the implementation of the honeypot software is possible in vehicular environment.

1.1. Motivation

As our common world is based to the huge extent on modern technologies like computers, Internet, and recently more and more developed wireless communication; it is highly open to wide range of threats. Intruders have many possibilities to interfere our lives causing much harm. All modern technologies have to be properly secured disabling intruders to obtain illegal access to its resources.

Implementation of wireless access into road conditions gives many possibilities to improve overall performance of traffic. However, being not secured properly can lead to disastrous effects, like putting health or life of user at risk, not saying about revealing confidential data, like credit cards number.

Motivation of this dissertation is to contribute in creation of powerful tool to improve the security system of WAVE and hence, contribute to increasing of safety on roads, decreasing number of data theft, accidents, deaths and any other road disturbances.

1.2. Objectives

The main objective of this dissertation is to study deeply the relatively new topic of WAVE technology in the VANET, focusing mainly on security aspects and honeypot implementation possibilities. Objectives of this dissertation can be enumerated in following way:

1. Study background of WAVE and VANET
2. Study honeypot software
3. Design architecture of Vehicular Honeypot
4. Chose the appropriate honeypot solution to be implemented

5. Propose necessary changes to make the implementation into WAVE possible

1.3. Dissertation Structure

The rest of the dissertation is organized as follows:

The second chapter is a background providing principals about WAVE, VANET and security oriented issues.

The third chapter is a honeypot case study - research about the honeypot software, its features and recent available solutions.

The fourth chapter is a presentation of main assumptions of Vehicular Honeypots followed later in the dissertation.

The fifth chapter is a deep study of honeyd – open source TCP/IP honeypot chosen to be implemented.

The sixth chapter is a core of the dissertation being a challenge to make the chosen honeypot software meet the requirements of WAVE.

The seventh chapter is a future work and conclusion being the outcome of this dissertation.

\

Chapter 2

Background

2.1 Intelligent Transportation System

Intelligent Transportation System (ITS) is an enterprise which aims to implement wide range of technologies (wired and wireless) into - generally speaking - transportation systems. It is designed to cope with most common problems of nowadays vehicular communication and additionally provides wide range of new opportunities, and what is even more important – improves safety.

Thanks to ITS, following improvements can be implemented into road infrastructure:

- Arterial Management
- Freeway Management
- Transit Management
- Traffic Incident Management
- Information Management

- Crash Prevention & Safety
- Emergency Management
- Commercial Vehicle Operations
- Road Weather Management
- Electronic Payment & pricing
- Intermodal Freight
- Roadway Operations & Maintenance
- Traveler Information

And into vehicles themselves:

- Collision Avoidance
- Driver Assistance
- Collision Notification[31]

Elements of ITS, its structure and all improvements are presented in below figure:

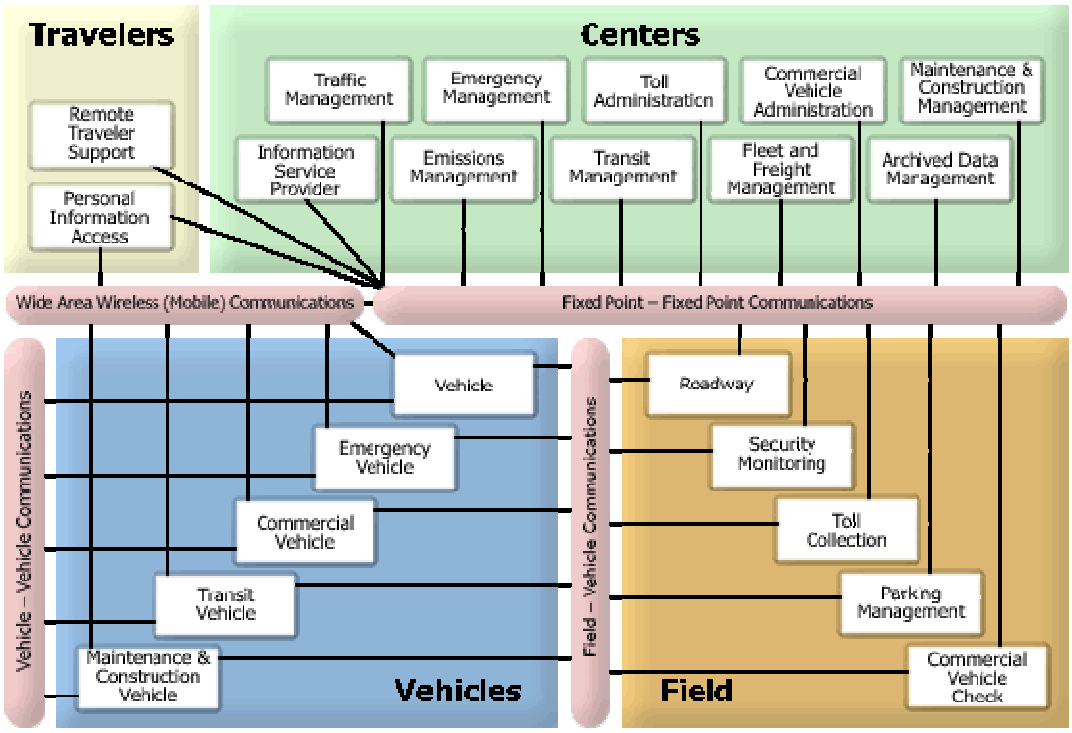

figure 1: Elements of ITS [31]

All of these improvements have a purpose to deal with the biggest problem in terms of road communication - traffic congestion. It is a common problem as being the product of technological development, motorization, and growing human population. ITS by efficient traffic managing also copes with second problem which is associated with urbanization - air pollution. By optimization of routes and decreasing congestion, it reduces fuel consumption, what results in reducing of gas production.[28]

ITS enhances safety on the roads, decreasing the number of accidents and deaths. It also reduces time of travel what is associated with reduction of traffic congestion.[26]

ITS addresses many modern technologies, however this dissertation is focused on wireless communications only

2.2 Vehicular Ad-hoc Networks

Vehicular Ad-hoc Networks (VANETs) are an instance of ITS. This technology forms mobile networks based on nodes which are either in motion or remain stationary. The main goal of VANET is putting into life all assumptions of ITS. Mutual communication between nodes (road users or infrastructure) provides fluent information exchange what contribute to increasing of road safety.

Infrastructure can also serve a role as a gateway to the Internet what additionally will, increase usability of this type of network – the time spent in traffic jam will be no more wasted [29].

2.2.1 Wireless Access In Vehicular Environment (WAVE) Technology

Communication in VANET is mainly based on WAVE technology – Wireless Access In Vehicular Environment. The WAVE system, based on a group of IEEE standards, is a radio communication system intended to provide seamless, interoperable services to transportation. These services include those recognized by the U.S. National Intelligent Transportation Systems Architecture and many others contemplated by the automotive and transportation infrastructure industries. These services include vehicle-to-roadside (V2R) as well as vehicle-to-vehicle (V2V) wireless communications.

WAVE standard family defines: architecture, model of communication, security system, physical access and management structure of VANET.

The general aim of WAVE is to **provide interoperable wireless networking services for transportation. [5]**
Additionally, it provides:

1. Traffic information:
    - whether condition,
    - road condition,
    - traffic congestion
    - location of road facilities (hotel, eating zones, gas stations)
2. Traffic warnings:
    - car accidents
    - unforeseen incidents on the road
    - local whether breakdown
    - sharp bends
    - illicit crossing the crossroad on the red light by other car
    - sudden breaking of the car in front of warning receiving car
    - lane change warnings

3. Possibility to pay charges (highway tolls, gas) automatically – tolling

4. Car service points with information about general technical condition of nearby traveling vehicles
5. Easy video – audio transmission
6. GPS maps actualization
7. Services, like police forces with possibility to
    - Maintain surveillance
    - Distribute speed limit warnings and pull-over commands

Taking into consideration the most appreciated advantage of WAVE and ITS - increasing safety on the roads, WAVE contribute to it by:

1. Immediately send warnings to all vehicles concerned about dangerous events
2. Taking control over vehicle by on board computer system in sudden and fast-reaction-needed situations

These all advantages can be fulfilled due to cooperation between vehicles and road infrastructure. V2V communication is an automobile technology designed to exchange of messages containing useful information between vehicles without participation of any other entity. V2R is oriented on gaining up-to-date information from roadside unit.

This cooperation makes possible a range of applications that rely on wireless communication between road users, including vehicle safety, public safety, commercial fleet management, tolling, and other operations

.

2.2.1.1 WAVE standards

WAVE technology is specifically described in following standards:

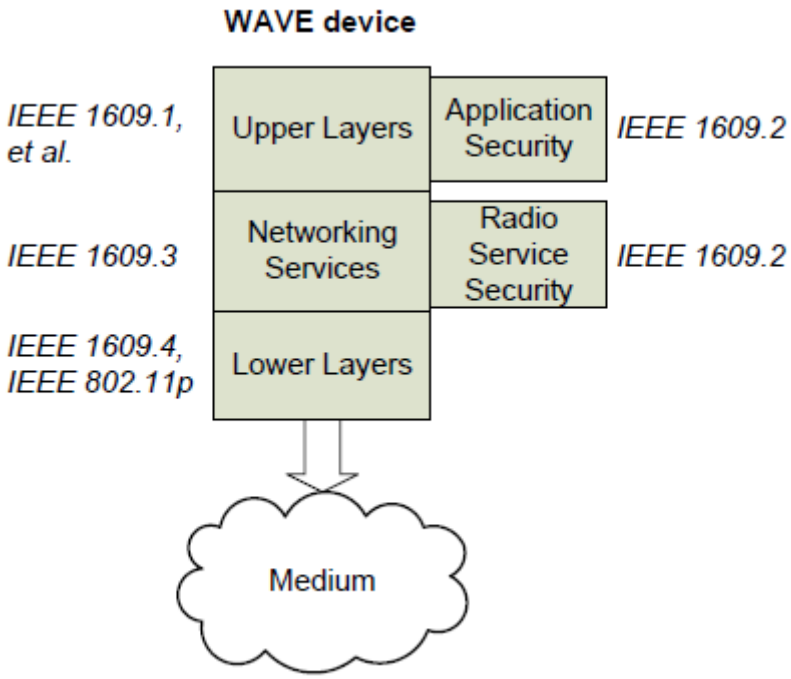

figure 2: WAVE Radio Stack [5]

IEEE 1609.1-2006 - Trial Use Standard for Wireless Access in Vehicular Environments (WAVE) – Resource Manager specifies a wireless access in vehicular environments (WAVE) dedicated short-range communications (DSRC) application, known as the WAVE resource manager (RM),designed to allow applications at remote sites to communicate with OBUs, through RSUs. The WAVE RM, acting like an application layer, multiplexes the communications of multiple remote applications, each communicating with multiple OBUs. The purpose of the communication is to conduct information interchange, needed to implement the requirements of the remote WAVE DSRC applications. [4]

IEEE 1609.2 -2006- Trial Use Standard for Wireless Access in Vehicular Environments (WAVE) - Security Services for Applications and

Management Messages defines secure message formats, and the processing of those secure messages, within the DSRC/WAVE system. The standard covers methods for securing WAVE management messages and application messages, with the exception of vehicle-originating safety messages. It also describes administrative functions necessary to support the core security functions. [5]

IEEE 1609.3 -2007 - Trial Use Standard for Wireless Access in Vehicular Environments (WAVE) - Networking Services defines network and transport layer services provided to WAVE device and systems, including addressing and routing, in support of secure WAVE data exchange. It represents roughly layers 3 and 4 of the OSI model and the IP, UDP, and TCP elements of the Internet model. It also defines Wave Short Messages, providing an efficient WAVE-specific alternative to IPv6 (Internet Protocol version 6) that can be directly supported by applications. Further, this standard defines the Management Information Base (MIB) for the WAVE protocol stack. [4] [30]

IEEE 1609.4 -2006- Trial Use Standard for Wireless Access in Vehicular Environments (WAVE) - Multi-Channel Operations provides enhancements to the IEEE 802.11 Media Access Control (MAC) and physical layer (PHYs) to support WAVE operations in context of multi-channel communication. Specifically, the multi-channel operation (channel coordination) provides mechanisms for prioritized access, channel routing and coordination, and data transmission. [30] [7]

Additionally, the WAVE 1609.X standards rely on **IEEE P802.11p – IEEE Draft Standard for Information Technology -Telecommunications and information exchange between systems - Local and metropolitan area networks - Specific requirements - Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) specifications Amendment 7: Wireless Access in Vehicular Environments**

This proposed standard specifies extensions to IEEE 802.11 MAC and PHY that are necessary to provide wireless communications in vehicular environment on the basis of Dedicated Short Range Communication.

The 802.11p amendment to IEEE 802.11 std. gives support to wireless local area network in vehicular environment. The main challenge of this standard is highly dynamic environment.

The first significant amendment introduced by IEEE 802.11p is the possibility of direct communication between two stations without necessity to belonging to any BSS (Basic Service Set). Those stations stay in the so-called “WAVE mode”. The frames being transmitted or received have to contain wildcard BSSID value. The STA (station) can transmit the data frame outside the BSS only if dot11OCBEnabled is set to ‘true’ (STA`s MIB - management information base – attribute within MLME - MAC Layer Management Entity). Otherwise, when dot11OCBEnabled attribute is set to ‘false’ or is not present in MIB – these situations are equal – the STA does not belong to any BSS. It allows for immediate data frames exchanges due to the fact that the authentication, association, or data confidentiality services are not used in this mode.

When dot11OCBEnabled is ‘true’, sending STA sets the BSSID field to the wildcard BSSID value. This mode is designed for safety messages in VANET environment which desires low latency and directness.

Next amendment which aims to improve performance of the environment is providing of WAVE BSS. It discards any authentication and security mechanisms leading to a reduced overhead for the WBSS setup. Such services (mostly security) are provided by upper layers as described in 1609.x std. family

Decision about joining the WBSS is made, based on the received beacon with all necessary information and configuration. The whole process of joining the WBSS is simplified basically by giving or not, a response to the WBSS advertisement.

Additionally, stations in WAVE mode cannot join infrastructure BSS or IBSS.

Termination of WBSS is dependent on a number of members, if it is none, WBSS finishes its existence. However, if the initiator of WBSS ceases its participation in WBSS, it can still exist.
Passive and active channel scanning is abandoned.

The PHY of 802.11p is based on 802.11 Orthogonal Frequency Division Multiplexing (OFDM) for the 5 GHz band with few enhancements. The PHY uses reduced clock/sampling rates – half clock mode with 10 MHz bandwidth what affects parameters like: bandwidth, carrier spacing, symbol length and frequency. It improves overall performance like making signal more robust against fading.

Following changes of course require changes in data packets frames and primitives as described in 802.11p amendment. [19] [8] [9]

2.2.1.2 Dedicated Short Range Communications (DSRC)

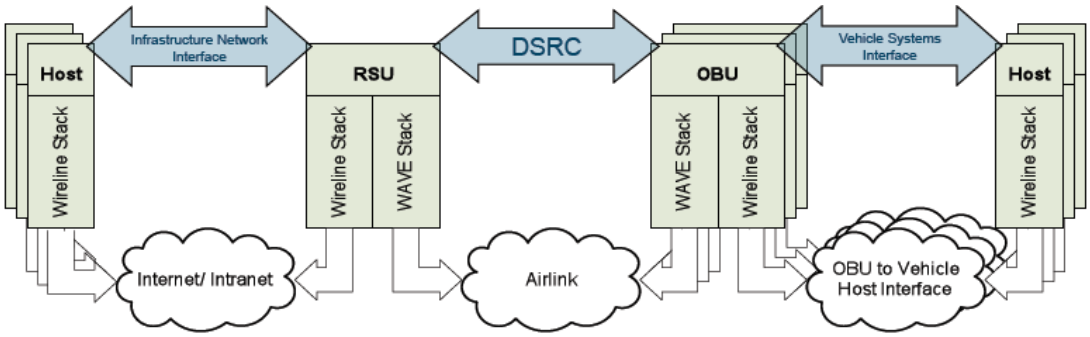

figure 3: DSRC System architecture [10]

DSRC is a technology which provides communication services for V2R and V2V operations. It is characterized by high data transfer rates and small latency.

DSRC uses a 5.9 GHz frequency with bandwidth of 75 MHz and an approximate range of 1000m. Vehicles can move with speed not exceeding 140 km/h.

In compliance with DSRC standard the dedicated frequency for V2V and V2R communication is in range of 5.855-5.925 GHZ. This band is divided into 7 channels, 10 MHz each. The first one (172) is dedicated for safety messages transmission and has the highest priority. [25]

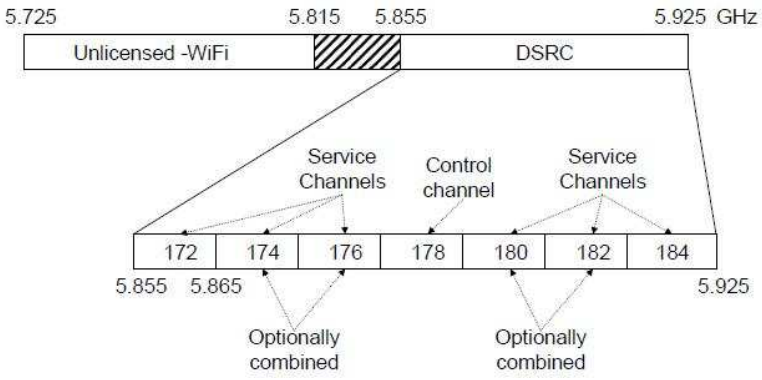

figure 4: DSRC Channel allocation [10]

The Bit Rate used within this standard widely differs depending on types of messages and range of communication. According to [20], Bit Rate used in the case of safety messages varies from 6Mb/s to 21 Mb/s. The maximum range for safety messages is more or less 300 meters. This length should be sufficient for drivers to take actions against the danger. According to [21], the time for proper reaction should be more or less 10 seconds.

Concerning highways and a car traveling with average speed of 120km/h, the distance of 300 meters takes the car about 9 seconds.

However, in densely jammed condition where cars are traveling closer to each other and slower there is no reason to maintain range of 300 meters. It decreases to 15 meters as only the closest vehicles would be interested in the event on the road.

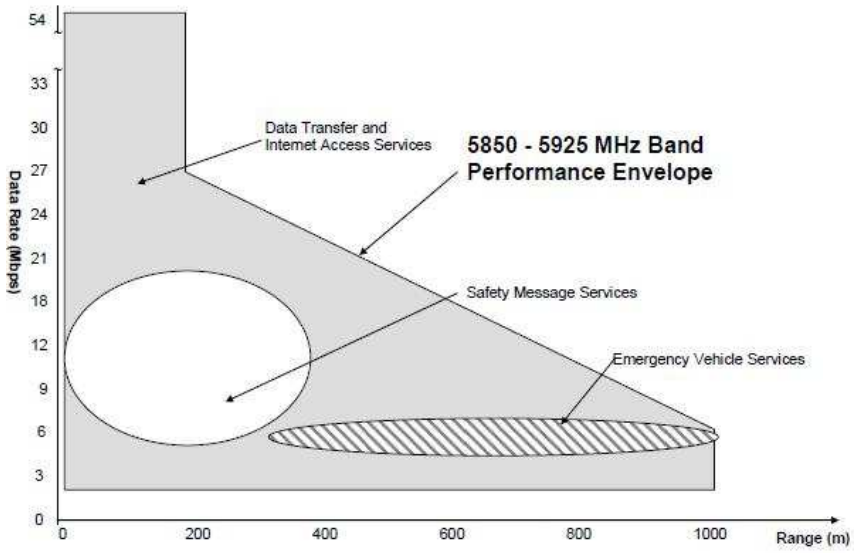

figure 5: DSRC Bit Rate and Range [20]

2.2.1.3 System Overview

Protocol Stack

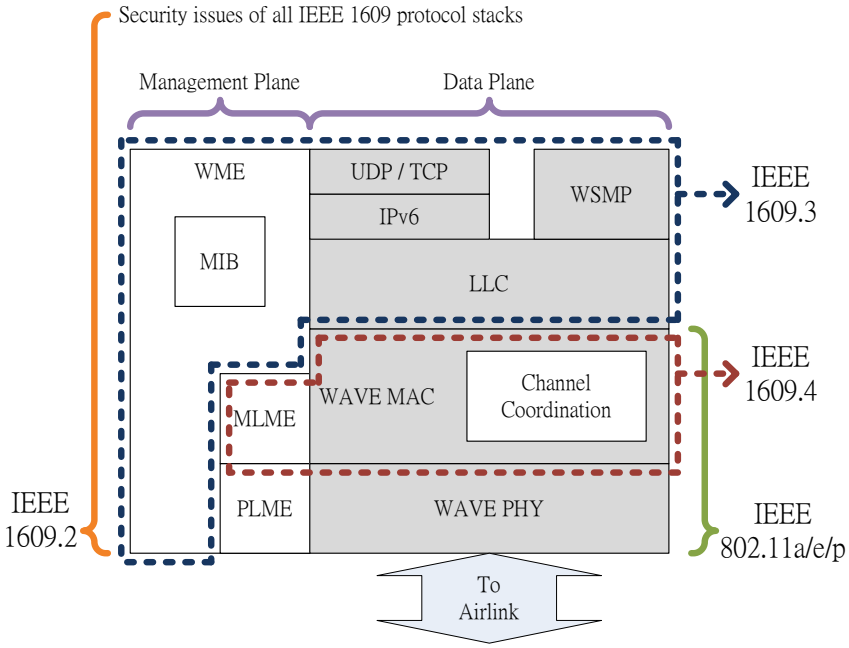

figure 6: Protocol Stack [32]

The stack consists of two planes:

Data plane – communications protocols – LLC, IPv6, UDP/TCP and WSMP (WAVE Short Message Protocol) – and hardware used for delivering data between applications and management entities.

Management plane – system configuration and functions providing management services. There exist specific entities for each of layers: PLME – Physical Layer Management Entity. MLME – MAC Layer Management Entity and WME – WAVE Management Entity which is treated as a more general collection of management services. The management plane uses data plane to passing management traffic. Management services include for example: application registration, WBSS registration, channel usage monitoring or IPv6 configuration.

Services provided by IEEE 1609.4 are used to enhance the performance of MAC layer by providing management of channel coordination and by supporting MAC service data unit (MSDU) delivery. It also take care of channel routing and user priority.

MLME and PLME management entities, MAC and PHY are widely described in 802.11 with enhancements introduced by IEEE 1609.4 and 802.11p standards.

WAVE MAC (802.11p) is designed to support two different protocol stacks IPv6 – delivered only on Service Channels and WAVE Short Message Protocol (WSMP) which can be sent on any channel and allows immediate exchange of information without necessity to belonging to any WBSS.

The WAVE networking services (IEEE 1609.3) provide LCC, network, and transport layer functions.

All communication is secured by the security system deploying mechanisms like authentication or certificates verification, as described in IEEE1609.2.

System Architecture

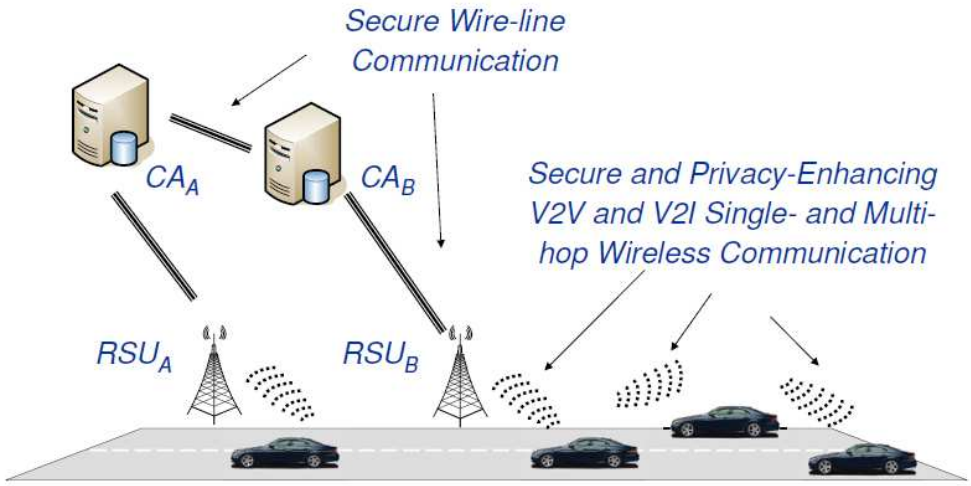

figure 7: System Architecture [11]

WAVE system involves many different types of entities. In day-to-day operations, the following entities are the most active:

Providers — Units that provide services on one or more service channels.

Users — Units that consume services offered by providers.

Roadside Units (RSUs) — Stationary WAVE devices located at the roadside. They support information exchange with OBUs. RSUs are typically embedded in infrastructure elements such as road signs and traffic signals; they may be moved from one site to another but do not operate when in motion. Providers will usually be RSUs. RSUs are licensed by site and may provide services on one or more service channels.

On-Board Units (OBUs) — WAVE devices that can operate when in motion. They support information exchange with RSUs and other OBUs. OBUs are typically embedded in vehicles though they may also be portable (hand-held). OBUs operate in private vehicles, and a major goal of the security protocols in his document is to enable them to operate without violating a driver's

reasonable assumptions about personal privacy. OBUs will usually be users, but may be providers in certain cases.

An important subset of OBUs consists of:

Public Safety On-Board Units (PSOBUs) — Network nodes embedded in public safety vehicles. These nodes are permitted by the relevant authorities to operate particular public safety applications such as traffic signal prioritization.

In addition, the following entities support security services:

Certificate Authorities (CAs) — Entities that are able to authorize other entities via the issuance and revocation of certificates. [5]

According to [5] CAs should be widely spread around some geographical terrain, being responsible for smaller regions each. The main role of CA is to manage the identity of nodes which are registered and authenticated in a given region. CA provides nodes with certificates (identity of owner and public key) or revokes them, as well as, distributes certificates of nodes which are moving into jurisdiction of other CA. Without this operation nodes managed by different CAs can not interact witch each other.

Communication Protocols

WAVE technology provides communication on behalf of two protocols stack: Internet Protocol version 6 (IPv6) and a remarkably new and designed for purpose of WAVE – WAVE Short Message Protocol. The IPv6 networking protocol is determined to work within Service Channel (SCH) while the WSMP has been designed to work within both SCH and Control Channels (CCH). However, the WSMP is designed restrictive to WAVE – aware devices. The uniqueness of this protocol is based on possibility of direct controlling the physical– layer parameters, such as: transmit power, data rate or channel number by the application residing on device. Additionally, within WSMP the

MAC address of the receiving destination is carried, including the possibility of a broadcast address. WSMP has been developed to meet the demands of highly dynamic environment. Hence, it is designed to decrease the overload and latency. It is reached by possibility of that protocol not to attending a WBSS while working on CCH Additionally, the new low – overhead packet format has been developed:

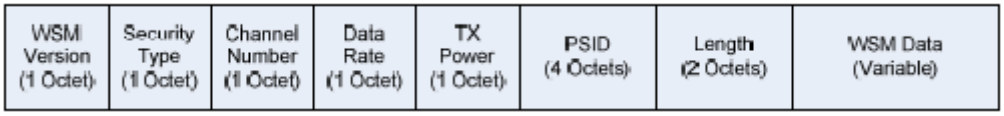

figure 8: WSMP packet [24]

WSM Version - is the number of the supported version of WSMP

Security Type - carries the information whether the packet is Unsecured, Signed or Encrypted.

Channel Number, Data Rate and TX Power - are used to direct control of parameters of physical layer.

PSID – Provider Service ID identifies the application that will process the WSM Data.

Length – informs receiving destination about the number of bytes in the WSM Data

In the case of WSMP the overhead is just 10 bytes.

In comparison, the IPv6 datagram overhead consists of IPv6 header and TCP or UDP header (more likely used in WAVE) .

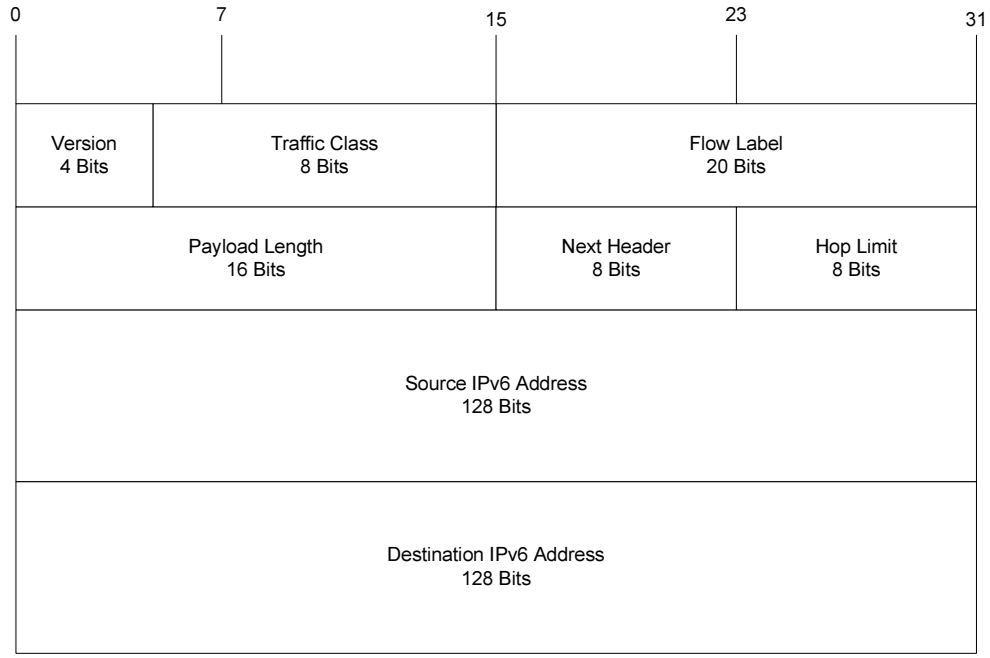

figure 9: IPv6 header

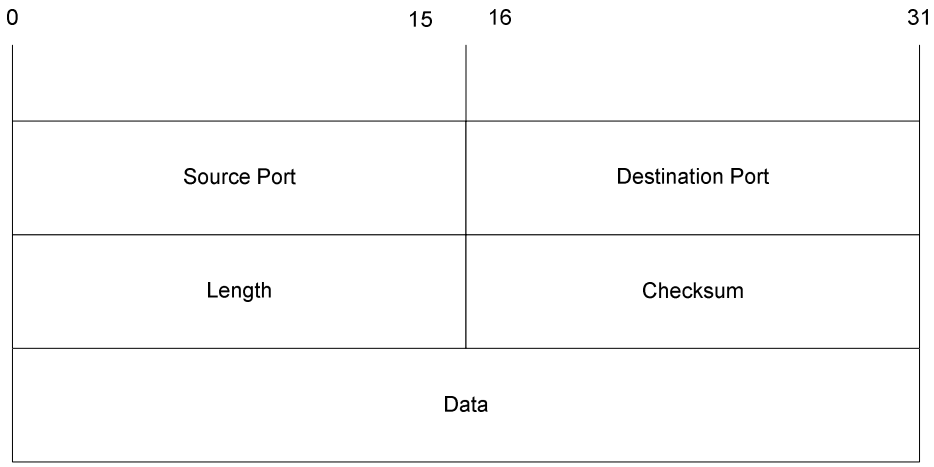

figure 10: UDP header

Together, overhead for UDP packet within IPv6 header is 48 bytes long.

Channel Types

WAVE distinguishes two types of communication channels: a single CCH and multiple SCHs. Commonly, WAVE devices operate on CCH, listening to high–priority applications and low latency WSMP system control messages. The device can switch to SCH if necessary. This is done, if device decides to join the WBSS. This enables the data exchange between applications located on devices associated to the same WBSS.

However, while transmitting data packets on the SCH, a device is still required to monitor the CCH channel in so-called 'control channel intervals' (CCH intervals). This is required because single channel devices (devices that can work only on one channel at a time) have to be able to operate high - priority safety or private service advertisements (send only on CCH) any time with high probability. WAVE devices shall continuously monitor the CCH when not synchronized to UTC (Coordinated Universal Time). [7]

WAVE Basic Service Set (WBSS)

WAVE devices can communicate with each other in two ways: in context of WBSS or directly to each other using the WSMP on the CCH. Association to WBSS enables applications to use either WSMs or IP traffic on the SCH associated with that WBSS.

WBSS is announced in the air and all devices with corresponding application set can decide to join it.

There are two types of WBSS: Persistent WBSS which is announced periodically what makes it suitable for providing general Internet access and non-persistent WBSS which is announced once, on WBSS initiation.

Threat model

VANET faces wide range of threats concerning malicious attacks.

In creating efficient and safe security system the awareness of every potential malicious attack together with their effects is a crucial thing.

1. Internal (ECU, internal components of car) [1] [3]

Targets

The targets of attacks are the internal components of car (Engine Control Module, Electronic Brake Control Module, etc. [18]

Desired results

Forcing abnormal behavior of components or taking control over them

Way to obtain access

1. By physically inserting a hacking device in the car and gaining access to internal network of vehicle via the ubiquitous OBD-II port (On Board Diagnostic port).
2. By wireless interface - OBU

Danger

Putting driver and passenger's life or health at risk

Attack Methodology

1. Packet Sniffing and Target Probing.
2. Fuzzing (Fault injection)
3. Reverse-Engineering

2. External – OBU and RSU

Target

Users of road – OBU and Road Side Units

Desired results

1. Forcing desired behavior of road users
2. Obtaining secret information
3. Spying

Way to obtain access

1. By wireless interface of OBU or RSU
2. Via the Internet
3. By physically inserting a hacking device in device and plugging it to OBU or RSU

Danger

Losing privacy and anonymity of users, traffic chaos

Attack Methodology

1. Internet originate attacks
2. Via wireless interface

a) Sybil Attack
b) Bogus Information
c) Denial of Service (DoS)
d) Impersonation (Masquerade)
e) Alteration Attack
f) Replay Attack
g) Illusion Attack

3. Messing with security ciphers and algorithms

a) Fault injection.

Generally, the IEEE 1609.2 [5] in Annex F provides classification of intruders who can jeopardize communication in the network:

- **Class 1**: Has a radio transmitter/receiver but no keying material.
- **Class 2**: Has a valid OBU or RSU.
- **Class 3**: Has extracted the keying material from an OBU or RSU, allowing them to either pretend to be that OBU or RSU in multiple locations simultaneously, or force the revocation of that keying material.
- **Class 4**: Insider at an organization with security administrative functions (HSM manufacturer, CA, vehicle manufacturer, government, etc.).

Every class of intruders has different threat possibilities. Class 1 equipped with radio transmitter is able to replay or tunnel the legitimate messages. Class 2 can broadcast fake messages or sabotage protocol stack connections. It can also be used to imitate fake services that could enable intruder to obtain secret information or track some device (in the case of RSUs the threat of being tracked can be omitted). Class 3 intruders with keying material can simulate physical device that can cause communication chaos. The biggest area of threat is the inside intruder which can gain access to many secret information like private data, keying material, certificates and so on.

All RSUs and OBUs have to be fully secured against all of these type of intruders because the effects of breaking into any node could be disastrous. Additionally, in contrast to OBU, RSUs create widespread, wired network connected to many remote entities like certificates centers or servers with Internet resources what cause that this network is much more vulnerable. The threat is based on taking control over Resource Manager within RSU which manages all communication (commands and data) flow between remote Resource Manager Applications and Resource Command Processor within OBU. What is more, without proper security systems, intruders once obtained illegal access to a RSU can possibly reach every part of the network and control even remote nodes of it. It can lead to terrible harm for entire network.

2.3 Security In Vehicular Communication

WAVE can open many opportunities and bring to life many services, however it is also prone to attacks. Implementing WAVE into everyday life, security issues have recently gained much consideration. Many solutions like cryptography or certificate issuing have been used but any of them are perfect and completely sufficient to cover all security – oriented problems. Despite using these solutions, many messages are still prone to being forged or replayed. IEEE 1609.2 std. [5] gives example of threats against the BasicSafetyMessages (messages conveying information about road conditions). They can be used to track a private car and unfortunately security solutions proposed in this standard does not address this threat

2.3.1 Aims of Security System

The vulnerability of many WAVE entities and confidentiality of most information and messages used or send by WAVE are sufficient reasons to maintain security system. Actions have to be undertaken to prevent the system from being attacked in different ways such as: message capture, eavesdropping, spoofing, alternation, spam or replay.

In addition, the highest possible privacy of users of the system has to be enforced. Security systems should enable to use system anonymously, without threat of being traced. The user should not be personally linked with the WAVE' messages.

Generally, the main aim of security system in WAVE is to provide:

1. Confidentiality

Provided by *encrypting* messages for a specific recipient in a specific way that only recipient can *decrypt* and read its content, what prevents from message capture or eavesdropping by third party.

2. Authenticality

Provided by confirmation of origin of the message, what prevents from accepting incorrect message content caused by altering the message during the transit.

3. Integrity

Provided by confirmation that a message has not been altered in transit, what prevents from accepting incorrect message contents caused by altering the message during the transit.

4. Anonymity

Provided by minimizing storage of a personal data which can uniquely identify vehicle, his owner or its position, what prevents from revealing vulnerable personal information.

What is even more important, it prevents association in the case of multiple messages.

2.3.2 Cryptographic Methods

The purpose of cryptographic algorithms is to encode messages, generally, to present then in a form that it can be readable only for the recipient of the message. The action of changing original message (called plaintext) into secure, encoded form (called ciphertext) is called *encrypting.* The reverse action, which enables a recipient of the message to decode the ciphertext to a readable version is called *decrypting.* The most common way to get the ciphertext is to use a special secret key. Usually the same key is used for decryption and that is why the key should be known only to two entities: sender and recipient.

This algorithm is called symmetric key ciphers. The algorithm, which implements two different keys is called asymmetric key (public key) cipher.

2.3.2.1 Symmetric Encryption

Symmetric encryption is the cryptographic mechanism based on usage of one key to both encrypting and decrypting. In one hand it is the biggest advantage of this cipher but on the other hand it is the main weakness.

This shared secret key, may be a subject to discovery by an adversary and must therefore be changed periodically. The generation of new keys, commonly carried out using a pseudo-random-number generator must be very carefully executed because, unless properly initialized, such generators may be not properly secured against key discovering. The new keys must then be distributed securely, preferably by using a more secure (but also more computationally intensive) asymmetric key cipher.

The whole process of encrypting and decrypting looks as follows:

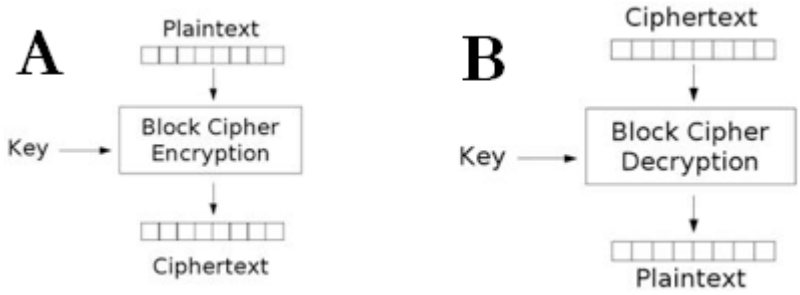

figure 11: Symmetric Encryption [34]

In a process, two and only two entities are concerned – **A** and **B**. Both know the same secret data, known as a *key*. To provide **confidentiality, A** uses the key to *encrypt* message, whereas entity **B**, who knows the same secret key, uses it to *decrypt* cipher text.

To provide **authenticity and integrity**, **A** uses the same or a different key to generate a cryptographic checksum or Message Integrity Check (MIC) on the message. **B**, who knows the same authentication key, can check the MIC. The MIC will only pass the check if the message and MIC have not been altered in transit.

A message may be encrypted only, authenticated only, or both encrypted and authenticated. In the latter case, it may be first encrypted and then authenticated, or first authenticated and then encrypted

The inner mechanism to “scramble” plain text into cipher text (Block Cipher Encryption) includes: Advanced Encryption Standard (AES). Firstly (to the 2002 year), mechanism called: Data Encryption Standard (DES) was followed.

DES uses 64-bit plaintext blocks and a 56-bit key, whereas AES uses 128-bit blocks and keys of size between 128 and 196 bits.

Longer secret keys are obviously more secure, but the size of the data block also plays a role in the security of the cipher. The DES cipher follows the

approach of Feistel which has aim to create ciphertext in 16 rounds of implementing Feistel function. Soon, it turned out that the security in DES mechanism, provided by the 56-bit key is weak. In effect, DES was replaced by AES mechanism which does not use a Feistel function. Instead, it is based on substitutions and permutations, with most of its calculations being finite-field operations.

AES Algorithm

AES algorithm uses blocks of 128-bit plaintext and implement three possible key sizes of 128, 192, or 256 bits. The higher number of bits in key implicate higher number of iterations:

Key lengths	Rounds
128	10
192	12
256	14

table 1: AES keys

AES encrypts all 128 bits in one iteration in contrast to DES where only 32 bits were encrypted. This is one reason why it has a comparably small number of rounds. The 128-bites block undergo an encryption process in a form of 4x4 array called the state (S with byte elements $s_{i,j}$ ($0 \leq i, j \leq 3$). As it is previously stated, whole array is transformed and modified in each of the iterations. The final cipher text is also of 128 bytes size. Each round of the encryption process, with exception of the first, consists of three so-called layers.

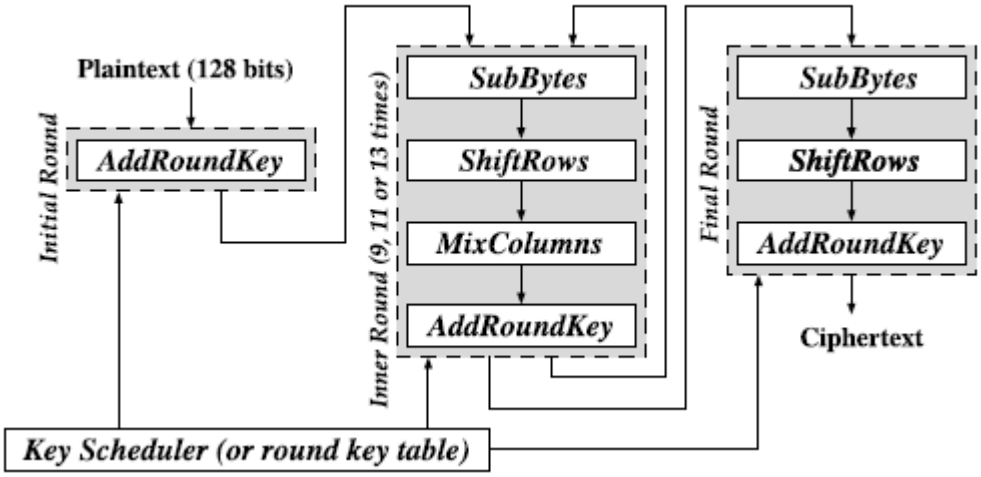

figure 12: AES Algorithm [2]

Key Addition layer **–** XOR operation between the state and round key obtained from Key Scheduler

Byte Substitution layer **–** Each byte of the state array undergoes a nonlinear transformation with 256 possible outcomes stored in lookup tables which has been designed to resist simple attacks. This introduces *confusion* to the data, i.e., it assures that changes in individual state bits propagate quickly across the data path.

Diffusion layer **–** It provides *diffusion* over all state bits and consists of two sub-layers: (both implementing linear operations):

- The ShiftRows layer permutes the data on a byte level.
- The MixColumn layer is a matrix operation which combines (mixes) blocks of four bytes.

In the first round only Key Addition is performed. In rounds 2-8 (for 128 byte size plaintext) Byte Substitution and Diffusion with sub-layers are performed. In the last round MixColumn operation is omitted which makes the encryption and decryption scheme symmetric.

The round subkeys are either generated on-the-fly following the key schedule process or are taken out of a lookup table that is filled up every time a new key is established. [2]

Problems of Symmetric Encryption

Concerning symmetric ciphers as one of encrypting methods in WAVE security systems we should mainly be focused on its drawbacks.

Firstly, conception of one, shared and secret key in the terms of using in WAVE has to be doomed to failure. It should be noticed that distribution of secret key between two users has to be done completely safe. In conditions of WAVE it seems impossible – the only way of efficient distribution of the key is over radio channel. To do that, additional encrypting mechanism should be implemented to ensure security what will cause that whole encrypting mechanism will turn to be ineffective. Secondly, symmetric encryption mechanism have to be limited only for message exchange in which any of sides can not take advantage of cheating.

Generally speaking, messages can be easily fabricated by one dishonest entity because it can not be in some cases verified whether the message was really sent by A or sent by B.

What is more, the biggest problem remains unsolved – the total number of keys. In system described above, every pair of entities has to obtain their own pair of same keys. In WAVE environment which is characterized by constant and rapid changes of communicating entities it is highly inefficient to obtain new pair of keys for every new communicating partner. It is easy predictable that on a two line highway with medium density of traffic, the number of keys needed to gain connection with every vehicle in range, will be enormous.

Additionally, the problem that may arise is the storage of those keys, but in this case it can be solved by short-time expiring keys. Unfortunately, the problem of huge “traffic” in channel of unsecured key data still remains.

2.3.2.2 Asymmetric Encryption

Asymmetric encryption, also called public key mechanism, unlike symmetric encryption uses two keys – public and private one. This relatively new approach to the matter of keys provides solution for problems which symmetric encryption cannot deal with. The communication algorithm looks as follows:

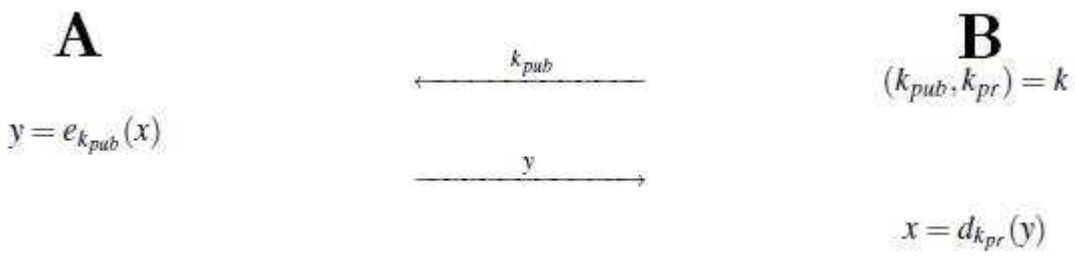

figure 13: Asymmetric Encryption [3]

Entity B (it can be RSU in WAVE) broadcast its public key – there is no need to be secret or hidden. All entities who want to communicate with RSU (it can be OBUs) encrypt their messages with provided public key, while an RSU after receiving the messages, decrypts them with its private key which is in strong, unbreakable relation (using one-way function) with the public key. There is one public key available to encrypt for everybody and one private for each to encrypt. This fact is very significant in relation to symmetric encryption.

However, it is still possible to use a symmetric encryption with secure transit of key through the channel. The solution is based on asymmetric encryption of secret symmetric key. The entity B is broadcasting public key. The entity A gain access to public key, generates its own secret, symmetric key which subsequently, is encrypted with a public key. The entity B is provided with asymmetrically encrypted symmetric key and with encrypted message **by AES mechanism**. It is very important because AES is much faster than the mechanism used in asymmetric encryption.

In asymmetric encryption to ensure that a message is **authentic and to provide integrity,** a cryptographic checksum is done. In the case of public key it is called **digital signature** in contrast to symmetric encryption in which case it is called MIC.

The sender of the message, in addition to public key encryption of the message, uses its own private key to create digital signature to prove that this message originated from it. The recipient obtains verification of the public key of the sender and the encrypted message. The next step is the verification process. The result of this test is positive whenever the message was not altered during transmission and the signature proves to be original.

The authentication by digital signature has one main advantage: it can be produced by only one entity, what makes the public key encryption excellent for WAVE in the terms of broadcasting messages. RSU has a possibility to provide services for large numbers of OBUs by distributing one public key and receiving digital signed messages from each one. With digital signature it is not possible to: forward signed messages, using them for other purposes or denying their origin. Additionally, it serves other purposes: it confirms that messages were send deliberately and ensures that messages cannot be altered without sender anticipation.

RSA Algorithm

RSA algorithm was the first to widely adopt asymmetric encryption algorithm. It was proposed in 1978 by Rivest, Shamir and Adleman. RSA algorithm is based on the use of two keys related by a one way function. RSA relies on **factoring numbers** (into small prime factors) that results from the product of large primes. Factorization is, as previously stated, a one way function. Its usability comes from the following relation:

Take into account two enough large primes, say, **72091** and **72101** (twin primes). The product of these two primes is: **5197833191**.

It was an easy way of following a one-way function. Making the reverse operation is much more difficult. It requires dividing that number by all primes from the beginning and checking whether the appropriate primes have been

found. In many cases it takes a lot of time and resources, and simply is not viable. The RSA algorithm takes advantage of this phenomenon.

RSA Key Generation

Entity **A** picks two large primes *p* and *q* in a way that both have similar length of bits but far away from each other in terms of value – to maximize the security of the key.

Calculate the one way function:

$$n = pq.$$

In the next step, compute Euler`s function for n:

$$\varphi(n) = (p-1)(q-1)$$

Simultaneously, random encrypting key – *e* is picked in a way that both – *e* and $\varphi(n)$ are relatively primes.

In the last step of algorithm the number *d* being inverse to $e \bmod \varphi(n))$ has to be found:

$$d = 1/e \bmod ((p-1)(q-1)).$$

Described mechanism provides entity A with two keys: **public key (e, *n***), and **private key** being defined as **(*d*, *n*).** The second one has to be kept in secret since it is used to decrypt messages whereas the first one, can be broadcast to as much entities as required.

The RSA modulus *n* should be at least 1024 bit long, which results in a bit length for *p* and *q* of 512.

RSA Encryption and Decryption

Encryption and Decryption are based on modular computations. Before encryption, firstly we have to divide plaintext into blocks, even though RSA is not a block cipher. It is necessary to encrypt blocks of smaller size than n due to binary value of plaintext.

After dividing plaintext and obtaining public key **(e, *n*)**, entity **A** is ready for encrypting as follows:

$$y = e_{kpub}(x) \equiv x^e \bmod n$$

Decryption by private key **(*d*, *n*),** processes respectively:

$$x = d_{kpr}(y) \equiv y^d \bmod n$$

All these operations make RSA algorithm extremely safe, very popular and widely used around the world. Unfortunately, RSA in comparison to AES requires much more time to be done and has bigger limitations concerning microcontroller implementation, both memory and power. What is more, it is sometimes desirable to establish more secure system (by increasing number of bits in key) what extends the computational overhead even more. In the borderline cases it simply turns to be inefficient. This is the reason why IEEE 1609.2 trail standard proposes the use of Elliptic Curve – Digital Signature Algorithm (EC – DSA) signatures.

2.3.2.3 Elliptic Curve – Digital Signature Algorithm (EC – DSA)

This type of algorithm used to produce digital signature is based on hard problems of Elliptic Curve Cryptography (ECC). ECC was discovered in 1985 by Victor Miller (IBM) and Neil Koblitz (University of Washington). It proved to have many advantages over previously described cryptography methods. It owns smaller computational overhead and needs shorter signatures and keys to

achieve the same security level. However, RSA algorithm is still much faster than ECC operations.

Symmetric Encryption	RSA and Diffie – Hellman	ECC
Key size in bits		
80	1024	160
112	2048	224
128	3072	256
192	7680	384
256	15360	512

table 2: ECC/AES/RSA keys comparison [4]

This significantly smaller length of keys and signatures in ECC is the result of using properties of the elliptic curve equation instead of the traditional method of generation as the product of very large prime numbers.

Jonathan Burr gives following definition of ECC [37]: An elliptic curve is a looping line intersecting two axes (lines on a graph used to indicate the position of a point). ECC is based on properties of a particular type of equation created from the mathematical group (a set of values for which operations can be performed on any two members of the group to produce a third member) derived from points where the line intersects the axes. Multiplying a point on the curve by a number will produce another point on the curve, but it is very difficult to find what number was used, even if you know the original point and the result. Equations based on elliptic curves have a characteristic that is very valuable for cryptography purposes: they are relatively easy to perform, and extremely difficult to reverse.

As it can be noticed, ECC gives relevant opportunities to improve performance of public – key cryptography thanks to its complexity and efficiency.

Key generation of EC – DSA

EC – DSA algorithm uses 3 steps to generate public and private keys:

1. Use an elliptic curve E with
- modulus p
- coefficients a and b
- a point A which generates a cyclic group of prime order q
2. Choose a random integer d with $0 < d < q$.
3. Compute $B = dA$.

The keys are now:
$k_{pub} = (p, a, b, q, A, B)$
$k_{pr} = (d)$

Signature Generation of EC – DSA

Every signature created by EC - DSA consists of a pair of integers (r, s), computing in following way:

1. Choose an integer as random ephemeral key k_E with $0 < k_E < q$.
2. Compute $R = k_E A$.
3. Let $r = x_R$.
4. Compute $s \equiv (h(x)+d\,r)k_E^{-1} \bmod q$.

Each value has the same bit length as q. In step 3 the x-coordinate of the point R is assigned to the variable r. The message x has to be hashed using the function h in order to compute s. [4]

Signature Verification of EC – DSA

1. Compute auxiliary value $w \equiv s^{-1} \bmod q$.
2. Compute auxiliary value $u_1 \equiv wh(x) \bmod q$.
3. Compute auxiliary value $u_2 \equiv wr \bmod q$.
4. Compute $P = u_1A+u_2B$.

5. The verification ver_{kpub} (x, (r, s)) follows from:

$$x_P \left\{ \begin{array}{l} = r \bmod q \Rightarrow valid \quad signature \\ \neq r \bmod q \Rightarrow invalid \quad signature \end{array} \right\}$$

The parameter x_P indicates the x-coordinate of the point P. The identity which receives the message with a signature accepts it only if the x_P parameter has the same value as the signature parameter r modulo q. [4]

2.3.3 Basic Threats to Cryptography Ciphers

This section presents a brief description of attacks on cipher algorithms.

The security mechanism, as it is commonly known, are designed to prevent hackers from obtaining keys and thus gaining access to private and valuable data. The main drawback of security systems is that they are mainly focused on hiding the cryptographic keys employing long time consuming computations of mathematical relations. Taking that fact into consideration, hackers have employed new approaches. They are more and more frequently replacing methods like brute force by methods which take advantage of side-channel information (a result of particular physical implementation of each algorithm). One example of such side–channel information can be time taken to execution an algorithm or a variation in the power consumed (in some algorithms the power consumption by implementing bit 1 or 0 of key differ noticeably). Generally speaking, side-channel attacks lead to abnormal behavior of algorithms by injection faults into a hardware implementation of a cipher. They are planned in a way that necessary information can be taken from the algorithm. The most common techniques of fault injection consist of: varying the power supply (generating a spike). changing the clock frequency (generating a glitch) or exposing the device to the impact of heat and intense light. The main advantage of this approach is “since most of these attacks induce transient faults, they allow the attacker to repeat his attempts multiple times until sufficient information is collected for extracting the secret key and even use the device after breaking the cipher” .[1]

To reach high effectiveness of such systems, extremely precise timing of fault injection has to be preserved. This can be done by examing the power and/or electromagnetic profile of the cryptographic device.

Taking into account symmetric ciphers the fault injection can occur during transfer secret key from EEPROM to memory in situation when encrypting or decrypting follows. Fault injection, in this case, is based on resetting every byte of the key and observing the outputs. After several trials the key is eventually discovered. A second type of side-channel attack is using a clock glitch to collapse the whole algorithm in the desired step (or round) of instructions. This method simplifies hugely key discovery because reveals the output of an algorithm only after some steps (or rounds) of encryption. Hence, the output is not secure completely.

Fault injection attacks on asymmetric algorithms, in contrast to symmetric can only be executed during decryption by flipping a random bit of the private key.

All above described techniques and many other dishonest behaviors may jeopardize the whole cryptographic system. To preserve system security, appropriate counteractions have to be considered. Over the years, many protection schemes have been proposed. Some of them can be called preventing (passive) operations and some of them active operations. First group includes schemes to prevent the disclosure of side-channel information like power consumption or timing. For example an additional pile of instructions are implemented into algorithm but without any significant role in encrypting. They have to be calculated what cause that the relation between bits of key and its real operational time or power consumption is blurred.
On the other hand, such actions increase delay in algorithm processing and cause power inefficiency.

The second group of schemes includes intelligent fault detection codes. The original designed cryptographic device has to be modified so that it will prevent attackers from obtaining output text after fault injection. One of such schemes is duplication of both, encrypting and decrypting process. It can be used assuming that the injected faults are transient and will not manifest themselves in exactly the same time in the two calculations. The other way, which is more efficient are Error Detection Codes (EDC). They are mainly

based on algorithm of comparisons between actual results of data encryption and predicted ones on respective steps of encryption.

2.3.4 Implementation of Security Protocols

The latest approach to secure system implementation is oriented to minimize the power consumption, processing time and dimensions of OBU device. The EEC mechanism (based on infeasibility of finding the discrete logarithm of a random elliptic curve element with respect to a publicly-known base point [33] has been used and following design of OBU device has been proposed in [12]:

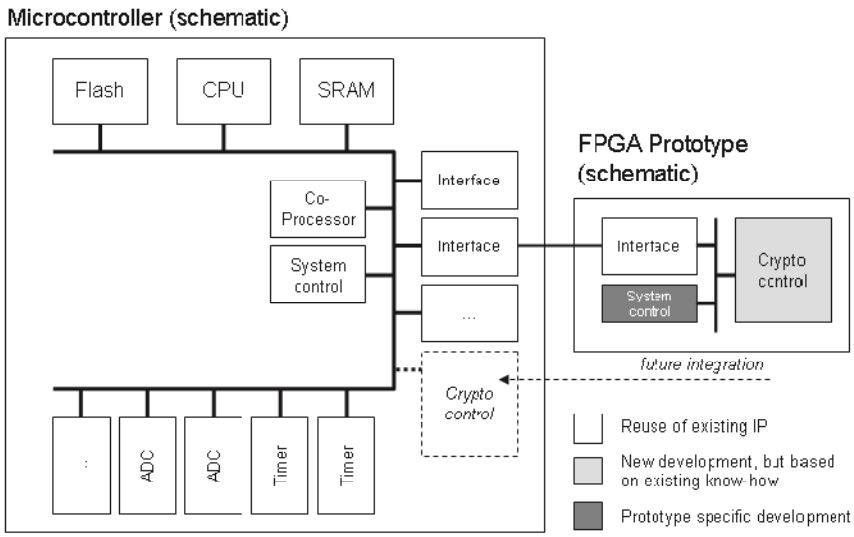

figure 14: Implementation of Security Protocols [12]

Implementation of any protocols in specific environment like WAVE has many difficulties which many seem to be unavoidable. First of all, the way of distributing OBUs should be planned. It has to be decided whether OBUs will be installed within production line of car manufacture and simultaneously certificates will be provided or the OBU device with appropriate certificate could be purchased and independently installed in car. Secondly, number of cars should be taken into consideration. As security protocols of WAVE are not fully

developed and all possible threats are not discovered yet, the possibility of modifying cryptographic system or any of OBU components should be maintained.

It should be planned in a way that modification could be easily carried out in large number of cars or nodes. The solution could be found in “hooking architecture” [13], which allows to modify security components without changes in any other part of the system. In the case of devices based on few modules, each responsible for different part of working system, only one module can undergo the required modification. In the proposed prototype [12] of integration whole components by using one microcontroller, modifications would not be so easy.

HSM (Hardware Security Module) is the subject of many concerns. Being responsible for all cryptographic operations, it should be fast, efficient, and resilient to attacks. One of the possible solutions is to implement HSM as an Application-Specific Integrated Circuit (ASIC) what is balanced solution connecting good security performance with low costs.

All above all, the real–time performance has to be preserved. The security hardware should be designed in a way that ensures the efficiency of whole system: balance between safety and performance.

However, in aim to increase a security in the network even more, the new innovative methods have to be implemented. One of them could be a solution, well known from computer’s networks, known as a honeypot

Chapter 3

Honeypot Case Study

3.1 Honeypot Explanation

A honeypot is computer software commonly used in IP-desktop systems to create a fake computer network environment, which pretends to be real and very often weakly secured. Hackers tempted by its poor security systems are trying to break into such network what is very closely monitored. Information obtained by analysis of intruder's attack patterns can be immensely useful in creating future, better security systems. Frank Spitzner provides the following definitions of honeypot [16]:

"A honeypot is an information system resource whose value lies in unauthorized or illicit use of that resource"

He also provides division into two main types of honeypots:

1. Low-interaction honeypots, which emulate real operating system with a wide range of services. This solution is very easy to deploy and has a very small risk of a honeypot being captured by the intruder. However, it collects limited amounts of data.

2. High – interaction honeypots, which provide real operating systems with services without emulation. They capture much more data and useful information by interaction with intruder. However, these honeypots require more careful implementation and configuration as this solution increases the risk of the usage of high interaction honeypot against other systems

Generally, honeypots have following functionalities:

- Creating fake local topology
- Creating controlled routing among local network
- Recording attempts of intruders

3.2 Value of Honeypot

Honeypots reveal tremendous and sometimes underestimated value for nowadays security systems. They can be used in many purposes: slowing down malicious software like scanning worms, confusing attackers or even deterring them from attacking a system or organization. However, the most important role of a honeypot is the detection of malicious behavior. Attacker detection and countermeasures are critical factors in the fight against intruders. Using nowadays tools it is very hard to respond quickly and efficiently to a failure in a system due to the fact that systems being compromised can not be analyzed offline. Even if it can be done, there still exists a problem of “data pollution” – it is extremely difficult to separate “normal user” activities from attempts of

intrusion. Honeypots provide solution to these problems. They are designed in a way that allows turning the honeypot into off-line state to be analyzed.

Additionally, all data stored by the honeypot is evidence of malicious and unauthorized activity what is immensely helpful. All this manners give an organization very useful and efficient countermeasures to respond rapidly to any form of intrusion.

Along with these previously described advantages of using honeypots, there are several more. Recording all intruders' activity or detecting malicious attempts give an overall look into tools, manners and technology of intruders. Such research gives opportunity to gain information that would be very difficult to be obtained in other ways. It is extremely helpful in understanding the intruders "way of doing", aims, methods and even motivation. All of these have enormous value for enhancing security systems.

3.3 Differences between Honeypots

At the beginning, the most common were honeypots used in case of local computer networks. They simulate hosts with operating system, servers, routers or any others devices available in the network. The huge progress in wireless communication reveals demand for honeypots simulating wireless devices – Honeyspots.

At this point, the main differences between 'wired' and 'wireless' honeypots will be introduced:

Criterion I: Role of a honeypot:

Wired:

- Confusing attackers by simulating network of many virtual hosts and recording any malicious attempts against them.

- Preventing attackers from obtaining access to data sources of LAN network

Wireless:

- Confusing attackers by simulating many access points.
- Recording any malicious attempts to compromise security system of wireless devices
- Preventing attackers from obtaining access to data sources of WAN network:
    - Attacks directed towards the wired network to which the wireless network connects. These attacks use the wireless network as a medium but the primary target is the network or information systems beyond it.
    - Attacks directed towards the wireless users. These attacks may use the wireless network as a medium to target the user's device wireless capabilities, and exploit the fact that the wireless device is enabled. The user may or may not be connected to a wireless network.
    - Attacks directed towards the wireless network infrastructure. These attacks focus on gaining control of the access point or wireless controllers, that is, the wireless infrastructure devices.

Criterion II: Network Topology

Wired:

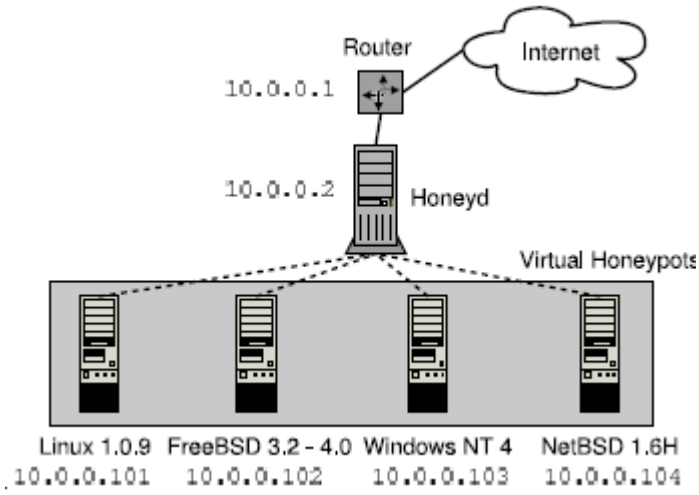

figure 15: Network Topology. On the example of honeyd [22]

Wired honeypot must be able to handle virtual honeypots on multiple IP addresses simultaneously, in order to populate the network with numerous virtual honeypots it simulates different operating systems and services. To increase the realism of the simulation, the framework must be able to simulate arbitrary network topologies. To simulate address spaces that are topologically dispersed and for load sharing, the framework also needs to support network tunneling.

A central machine intercepts network traffic sent to the IP addresses of the configured honeypots and simulates their responses.

Wireless:

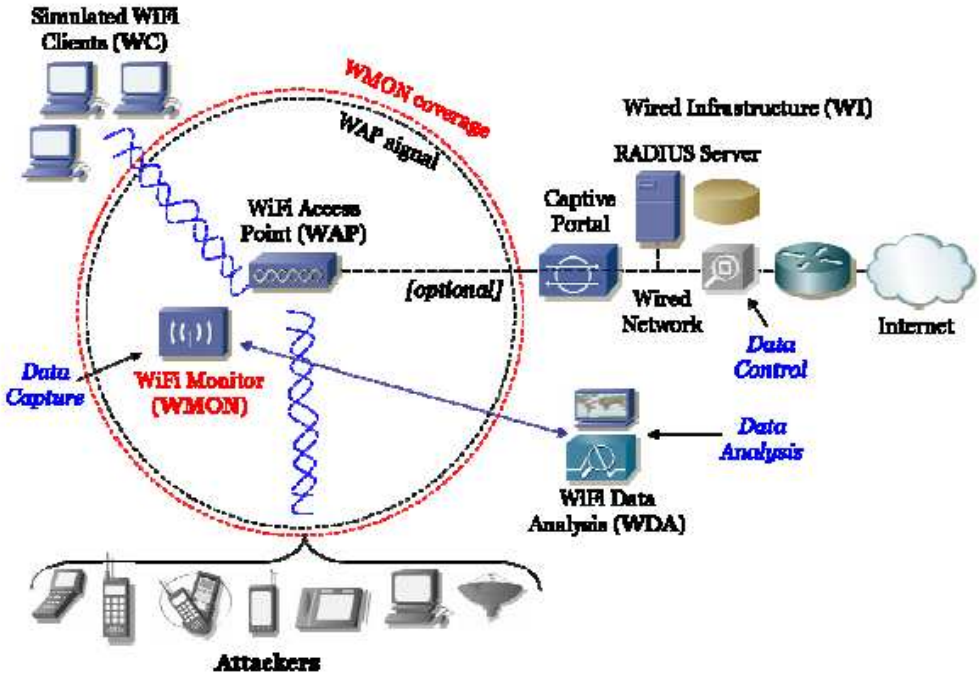

figure 16: Network Topology. On the example of HoneySpot [14]

A Wireless Honeypot has to combine requirements of traditional server honeypots with the requirements of the recent client honeypots research, or honeyclients. It must provide the wireless infrastructure plus the client capabilities required to simulate the presence of wireless clients in the wireless network.

Criterion III: Architecture

Wired:

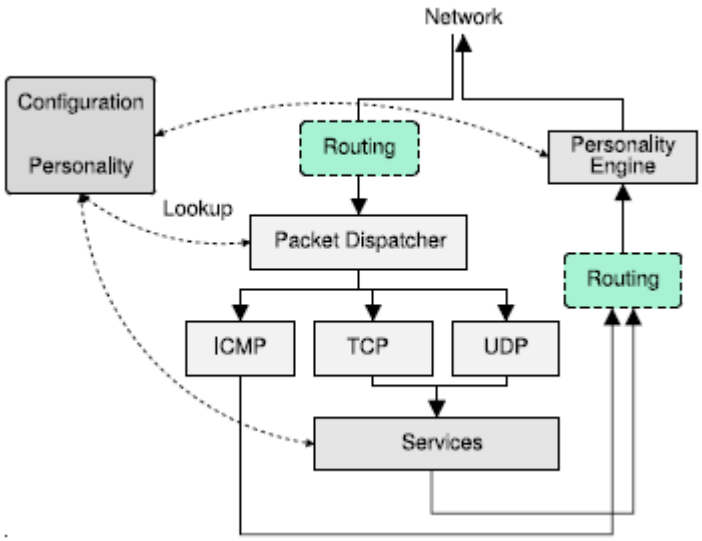

figure 17: Wired Honeypot Architecture. On the example of honeyd [22]

Incoming packets are processed by Packet Dispatcher and then send the correct protocol handler. For TCP and UDP protocols, packets reach appropriate hosts and services basing on IP addresses and port numbers retrieved from packet's headers. Previously configured services provide responses. All outgoing packets are modified by the personality engine to give illusion of network stack of real OS. The routing component is optional and used only when the honeypot simulates routing or network topologies.

Wireless:

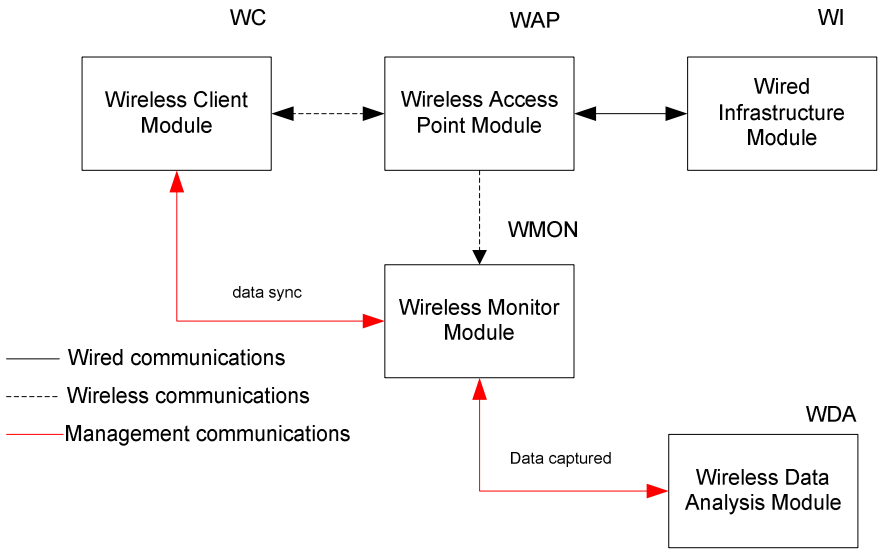

figure 18: Wirelss Honeypot Architecture. On the example of honeySpot [14]

The biggest difference between wired and wireless Honepot is wireless access module, which include WC, WAP, WMON, WDA. Wired Infrastructure Module can be simulated by wired honeypot.

Additionally, wireless honeypots are prone to a higher range of attacks due to wireless nature of obtaining access to network. Whereas in the context of wired network, attacker has to be connected into the network, in the wireless environment it has to be only in the range of signal of wireless device. These attacks are mainly based on vulnerabilities of authentication mechanisms (mainly WEP, WPA/2 or 802.1X/EAP) or other security mechanisms (MAC address filtering, turned off SSID broadcast)

3.4 Recent TCP/IP Solutions

3.4.1 Honeyd

Many open source honeypots can be currently found in the Internet which can be downloaded and modified by everybody. Much recognition was given to software called “honeyd”. This program accompanied by “farpd” can simulate computer network mixing IPs of real computers with fake IPs. Intruder pinging the random IP in the local network will receive response like from a real host. All IPs from the pool seem to be real hosts, confusing attackers. Farpd is a daemon that listens to ARP requests and answers for IP addresses that are unallocated. Using farpd in conjunction with honeyd, it is possible to populate the unallocated address space in a production network with virtual honeypots.

Additionally, this software is resistant to tools such nmap because honeyd also emulates any operating system at the network stack level. Intruders using this tool will receive bogus information that has been previously created. Using ready scripts it can be also possible to emulate many server’ services like FTP or SMTP.

Honeyd also can be used to emulate fake access point for wireless environment by changing mode of wireless card into hostap and creating and configuring DHCP server. Hostap is a driver for linux, working with wireless cards based on prism chipset. It adds more IEEE 802.11 management functions in the host computer and acts as an access point.

Example of Configuration (.config):

```
create windows
set windows personality "Microsoft Windows XP Professional SP1"
set windows default tcp action reset
set windows default udp action reset
set windows default icmp action open
add windows tcp port 139 open
add windows udp port 137 open
add windows tcp port 137 open
bind 192.168.1.140 windows

create server
set server personality "Astaro Security Linux 4 (Kernel 2.4.19)"
add server tcp port 21 "./ftp.sh"
add server tcp port 25 "./smtp.sh"
set server default tcp action reset
bind 192.168.1.141 server
```

3.4.2 FakeAP

Second software that can be useful is a representative of wireless honeypots. It is FakeAP, created by Black Alchemy Project.

The program has the ability to generate "*thousands of counterfeit 802.11b access points. Hide in plain sight amongst Fake AP's cacophony of beacon frames. As part of a honeypot or as an instrument of your site security plan, Fake AP confuses wardrivers, net stumblers, script kiddies, and other undesirables*". [35]

Chapter 4

Vehicular Honeypot - Assumptions

With fast development of VANET the possibility of deploying of honeypots in vehicular systems seems to be very tempting way to immune WAVE security systems against intruders. [17]

Whereas the security system described in the IEEE 1609.2 standard prevents from malicious attacks, like spoofing, eavesdropping or repetition, it does not completely deal with the problem. It fights with results of attacks but not with attackers. This is the role of a vehicular honeypot, which could work as a decoy for attackers who, for example, may want to hack security algorithm. Honeypots would improve their safety by revealing attacker`s tools, strategies or even its identity.

In this section, main assumptions concerning Vehicular Honeypot are presented.

4.1 Aims of Vehicular Honeypots

The main purpose, similarly to previous usages, is to simulate node of real network - OBU, RSU, CA (mostly some part of them) to reveal all malicious attacks and hence, improve security of network by gaining knowledge about attackers (frequent aims, tools or strategy).

4.2 Main Features of Vehicular Honeypots

- Have to be indistinguishable from other nodes and protected from exposure and from physical stealing
- Have to be able to interact with intruder (or normal users) or generate and simulate random traffic (messages) in way that external observer will not be able to found out that the traffic is fake.
- Be able to analyze the incoming traffic, assuming that the incoming traffic are not only malicious attempts.
- Should own full functionality as regular entity

Aforementioned conditions could satisfy the design presented in the next section.

4.3 Design

The honeypot solution can be used in vehicular environment in two ways: imitating RSU or imitating OBU. Designs of both devices are presented below:

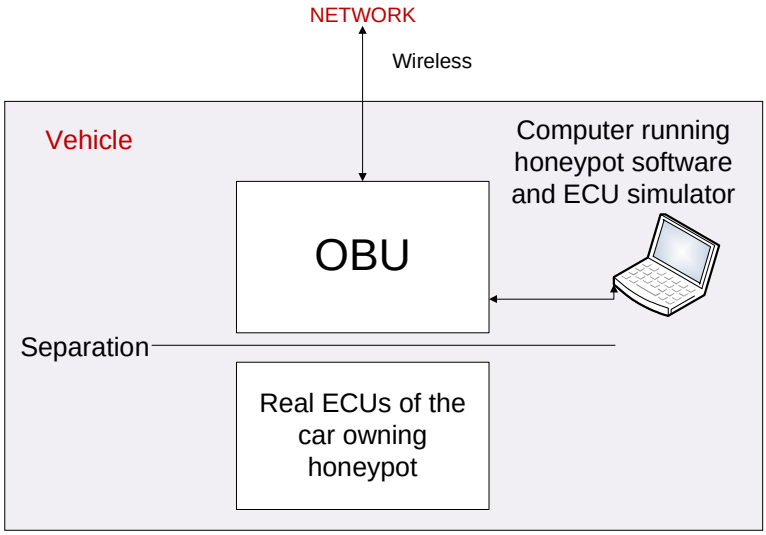

figure 19: Design of OBU Honeypot

The vehicle serving as OBU honeypot should be equipped with computer running Linux OS and honeypot software adapted to WAVE environment. The computer should be connected to the OBU, e.g., via a USB interface, so it can communicate with the IEEE 802.11p. As the vehicle Electronic Control Units (ECUs) are the targets of possible intruders attacks and, additionally, OBU honeypot would be prone to wide range of attacks, all ECUs should be separated from the wireless interface. It should be done due to the fact that any intruder's interference with ECUs of vehicle being the OBU honeypot could put driver's safety at risk.

Unfortunately, separation between OBU and ECUs narrows widely functionality of vehicle with OBU as a node of VANET. The purpose of connecting ECUs of a car with wireless interface is to broadcast messages about sudden abnormal behavior of a car, giving a possibility for other vehicles to react accordingly. With separation it is impossible, what is more, it can even lead to revealing OBU honeypot identity. The intruder by observing OBU honeypot vehicle could possibly notice, that this particular vehicle is not broadcasting any messages about events like sudden breaking. Although, there is a solution to this problem. The computer running honeypot should be also equipped with ECUs simulator software that could broadcast normal ECU`s messages in any desired time. It would give an illusion of realism.

The design of RSU honepot looks as follows:

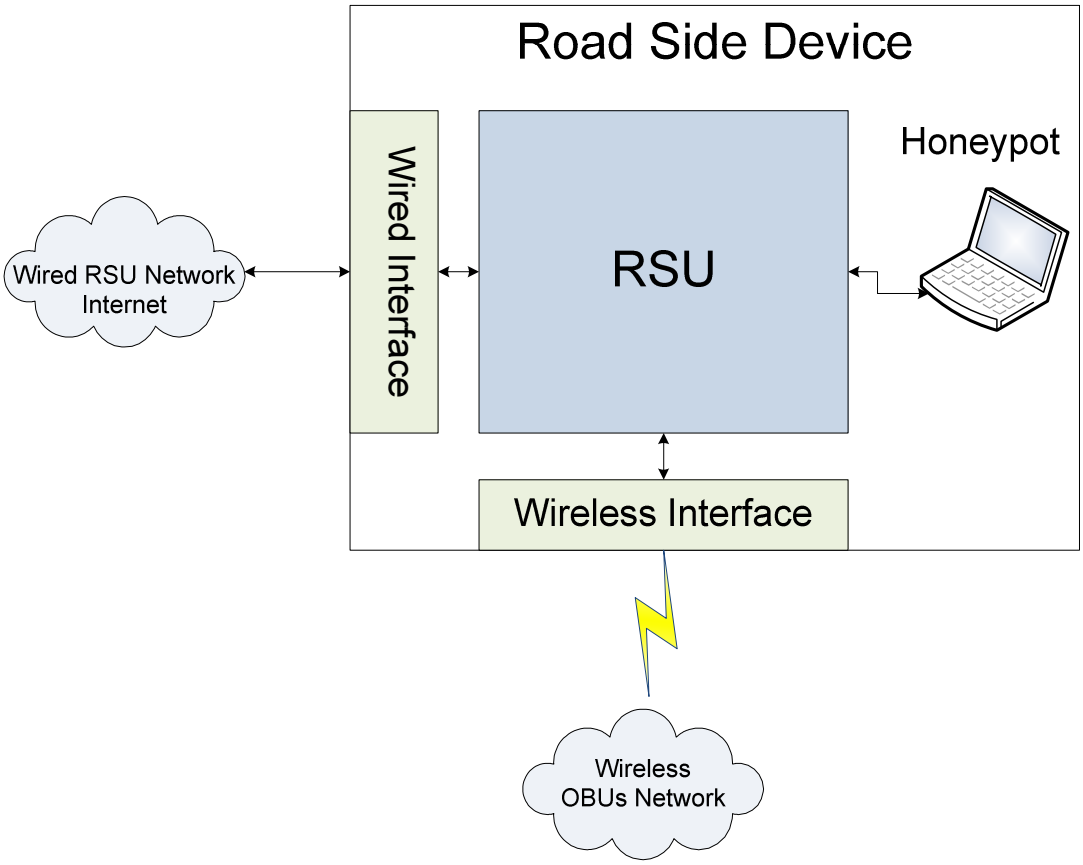

figure 20: Design of RSU Honeypot

The design of RSU Honeypot can be very similar to that installed in OBU. It should include computer running Honeypot software and software simulating all functionalities of RSU. In this situation RSU owns two gateways. One for wireless communication with OBUs and second one – wired used to inter RSU communication.

4.4 Functionality

Concerning implementation of honeypot software in conditions quite different than TCP/IP, changes in its functionality have to be considered. As it

was mentioned in chapter 3 honeypot in TCP/IP condition simulates topology and fake systems. It works in the range of the local network. Additionally it monitors traffic in specified IP range. However, the functionality of vehicular honeypot could be narrowed to that presented on below diagram:

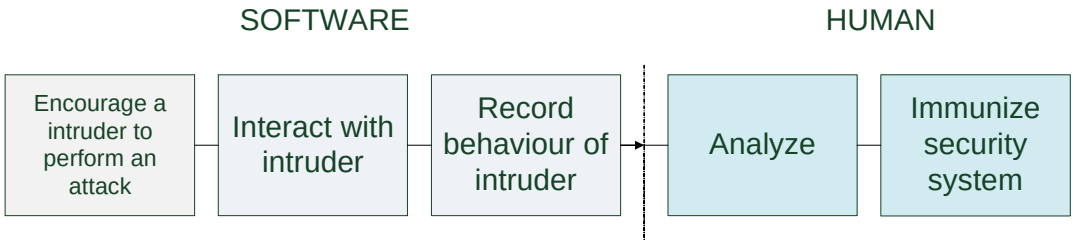

figure 21: Functionality of Vehicular Honeypot

The vehicle with installed honeypot software (OBU honeypot) would drive around some area, simulating an application, gathering information and monitoring every unauthorized attempt to attack its resources. The range of RSU honeypot would be much more smaller due to the fact that it would be stationary, covering only some area. Either OBU honeypot or RSU honeypot would imitate a regular node, attracting potential intruders and recording any interaction with them.

However, there are some doubts concerning vehicular honeypots.

First of all, the placing and number of such devices is matter of consideration. Probably, it would depend on intruder's activity and budget of authorities. What is more, in the case of TCP/IP honeypot, it has no productive value, and every interaction with it is perceived as malicious. To make it clear: any legitimate user does not have any necessity to interact with it. In vehicular environment it is not so obvious because of the specificity of VANET environment. OBU honeypot cannot be isolated from other users, because, as in the case of every ad-hoc network, every node is significant part of the network. Additionally, a honeypot will not be available for every user but only for some authorities which take care about the security system. In this way, a car serving as OBU honeypot has to attract intruders' attention to perform an attack. Hence, the way of interacting of honeypot with network and intruders

has to be considered. Two proposals of using OBU honeypot have been considered:

Fist one, owning real and attractive application interacting with users residing in OBU. In this context, honeypot would be used to capture and store on laptop hard disk all packets in and out coming from OBU. This would help to catch the malicious software like worms and additionally, the attractiveness of application would invite malicious behavior of intruders what would be desirable to reveal their methods and tools. However, giving possibility an intruder to interact with real application can lead to disastrous effect before any appropriate defensive action can be done.

A second option is providing a honeypot software with script imitating an application similarly to a TCP/IP usage. Honeypot would be capturing and storing all packets as in first option. Additionally, it would give an intruder the possibility to interact with fake application revealing his tools and methods. It could be very safe and convenient. However, this situation could mislead every user that would like to interact with the application because it will not be fully functional. To provoke an intruder to perform an attack, the application has to be advertised within WSA. So, it would be available to every one in a radio range.

Generally, it has to be a real OBU or RSU with all communication functionalities. Additionally, it has to own possibilities to capture, recognize and store any in- and out- coming traffic.

There is no place for fake topology or routing. There is no local network. The aim of vehicular honeypot would rather be gathering information about attackers` hacking techniques, than hiding the device or confusing intruders. Vehicular honeypot has to be designed and configured in a way that gives impression of interaction with regular OBU or RSU. This can be done by packets produced by honeypot imitating responses from an application or by simulation of ECUs behavior.

The honeypot software simulating application layer of an OBU has to been ‘placed’ in the WAVE suite of OBU to provide smooth flow of packets. All messages has to be directed to honeypot instead of application layer of an OBU

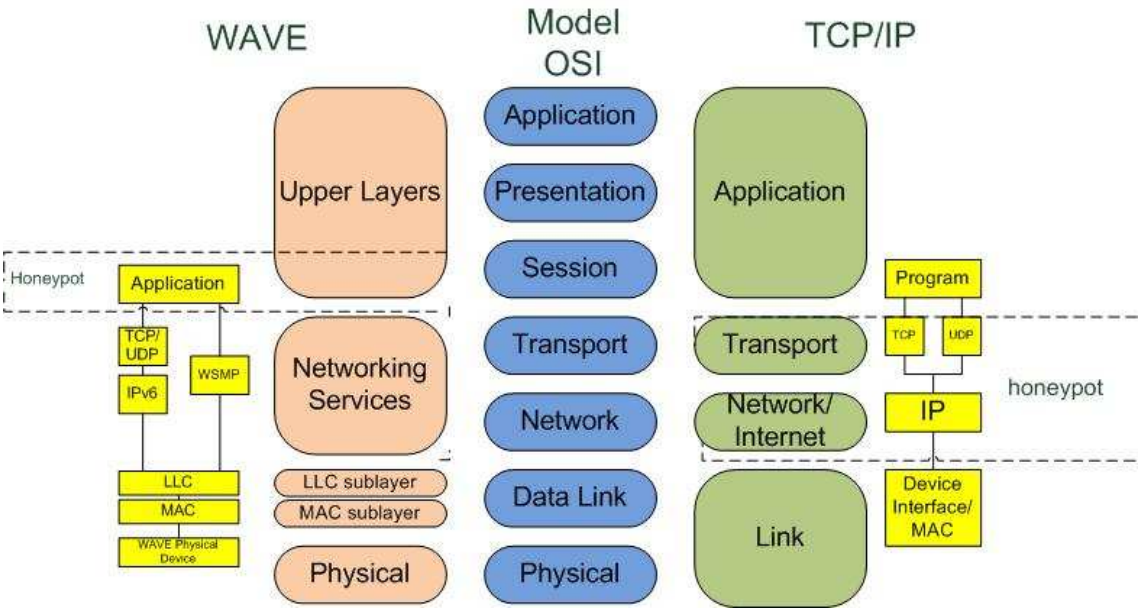

figure 22: Model Comparison

However, none of the aforementioned cases of usage of vehicular honeypot secures OBU or RSU from being hacked. In TCP/IP the honeypot creates wide range of fake hosts and services that intruder can interact with. This is safe because even if an intruder manages to hack into such system it will make no harm. In WAVE it will be highly inconvenient to populate a radio with fake entities due to highly dynamic environment. For that reason a driver or operator of OBU honeypot has to have possibilities to react rapidly to any effects of OBU hacking

Additional feature that would attract intruders to perform an attack can be badly configured security system or WBSS configuration but this solution needs very effective and rapid countermeasure too.

4.5 Interaction Model

The role of OBU honeypot is to provoke and capture malicious behaviour. To provide this functionality an intruder has to have the possibility to

interact with the honeypot. An intruder should be tempted to undertake malicious actions against OBU honeypot. It can be done, as aforementioned, by providing OBU honeypot with real and attractive application residing in OBU, in application layer or by fake, simulated by honeypot, residing on laptop.

In the case of real application, providing full functionality, the communication pattern looks as follows:

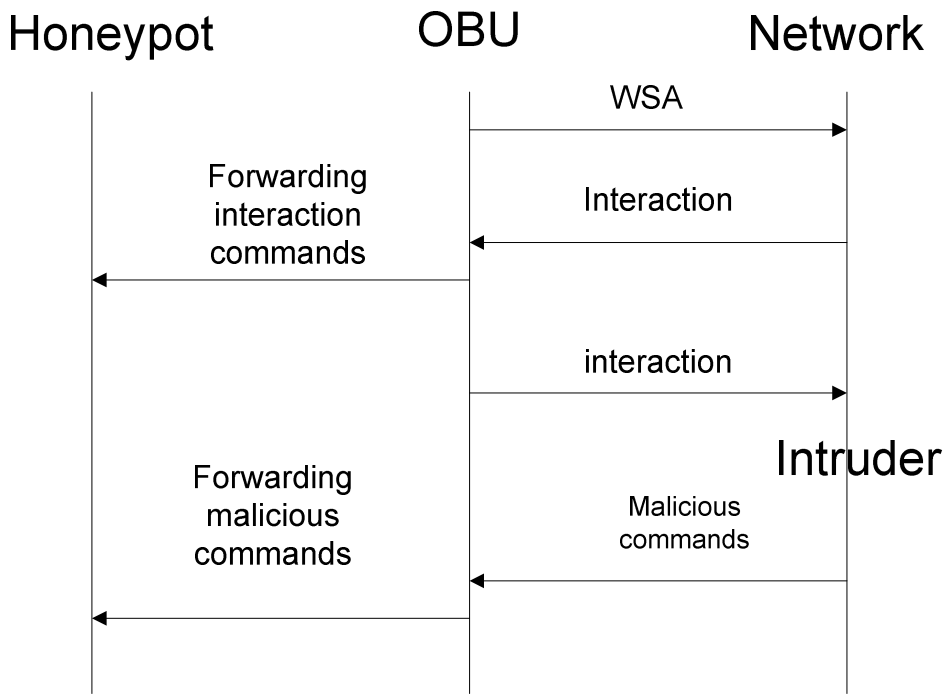

figure 23: Interaction Model I

The application residing within OBU is widely advertised and available for every user in a range. This application interacts with users normally, as specified in IEEE 1609.X standard family. The role of honeypot is narrowed to retrieve all traffic from OBU, log it and store.

A honeypot captures packets from USB interface. Then, they are processed, logged and stored on hard disk of an laptop.

In the case of a simulated application by honeypot the communication pattern differs slightly and is more similar to functionality of TCP/IP honeypot:

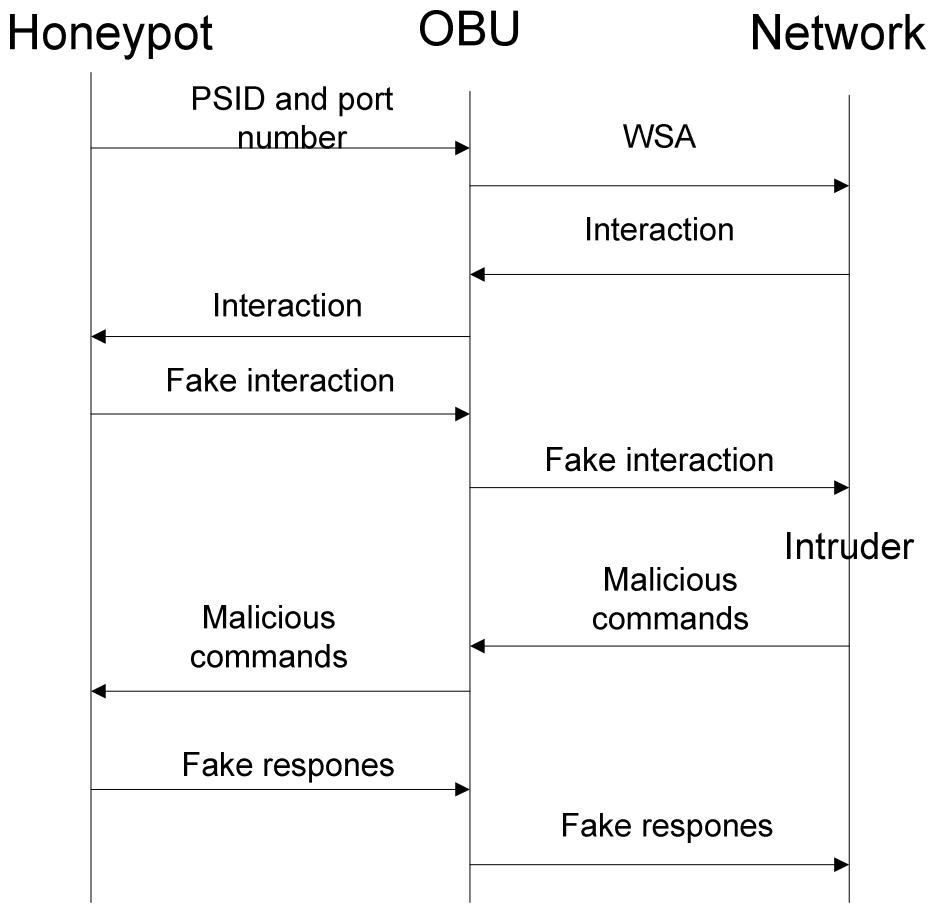

figure 24: Interaction Model II

Honeypot is equipped with fake application, which pretends to be a real, residing in the application layer of an OBU device. The appropriate script should be written, depending on a range of functionality of an application. Honeypot sends a PSID and port number of application to OBU, which produces a WSA message which is then broadcast. All processing of packets exchange or WBSS initialization and generally, all functionality described in IEEE 1609.X standard family should be provided by an OBU. However, all packets destined to application reach honeypot which responses to them to some extend, depending on application's script complexity. In the case of malicious commands, a honeypot can still provide an intruder with fake responses giving an illusion of interacting with real application of OBU.

Assumptions concerning vehicular honeypots described above give details about theirs implementation in VANET. Additionally, proper honeypot solution should be chosen to be used as a laptop software. Via the Internet research one certain was found which seemed to be able to adapt to VANET requirements. This software was honeyd. It was mainly chosen due to the fact that it proved to have every advantage of honeypot solution and owns a possibility of association a service script which would imitate some application. The next chapter is a deep analysis of this software.

Chapter 5

Honeyd Study

5.1 Introduction

Honeyd is a low-interaction, Uniux destined, open source honeypot created and maintained by Niels Provos, released in April 2002. It simulates virtual computer systems at the network level, having capability to monitor millions of IP addresses and allowing to populate unallocated IP addresses. It creates fake topology with configurable characteristic or routing, if necessary. It provides also a solution to deceive port scanners or operating system detectors. Honeyd aims to simulate TCP and UDP services and to respond to any network packets whose destination IP address is among those simulated by honeypots. Internet Control Message Protocol (ICMP) is also supported. Honeyd is highly customized by user to fit the requirements of local requirements [22] [23]

5.2 Architecture

The architecture of Niels Provos's honeyd consists mainly of 5 modules:

- A Configuration Database
- A Central Packet Dispatcher,
- Protocol Handlers (TCP, UDP, ICMP)
- A Personality Engine,
- A Routing Module

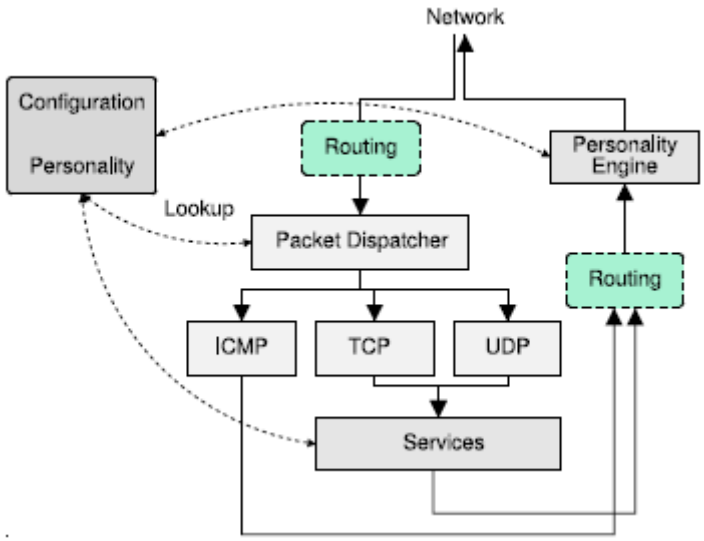

Figure 25: Architecture of Honeyd [22]

Packet Dispatcher

The Packet Dispatcher is the most important component. It provides processing of any packet that reaches honeyd. It is mainly focused on packets from three major protocols: TCP, UDP and ICMP. Incoming packets for other protocols are not processed and are discarded previously being logged.

The Packet Dispatcher firstly, recognizes the type of packet by checking its length and verifying the checksum of the frame. Before further processing,

this module retrieves, from the configuration database, the configuration that fits the destination IP address requirements of the packet. If compliance is not found, a default configuration template is used. After assignment of configuration template to the packet, it is delivered to the specific protocol handler.

Protocol Handler

Honeyd supports mainly 3 types of protocols: TCP, UDP and ICMP.
The ICMP protocol handler supports most ICMP requests. By default, all honeypot configurations respond to echo requests and process destination of unreachable messages. Any other responses are processed in a way specified in the configuration file. Generally the Protocol Handler process the received frame and produces the responding packet using, when specified in configuration file, the service scripts. Honeyd supports the three-way handshake method used to establish TCP connection and tear down TCP socket connections over the network.

UDP pockets are processed directly in the application. When a UDP packet is destined to a preconfigured closed port, an ICMP port unreachable message is sent to the sender, unless differently stated in the configuration personality.

Personality Engine

After being processed by the Protocol Handler the responding packet has to be subjected to changes by the Personality Engine to make them appear as real to an adversary probe.

As stated previously the Honeyd simulates the network stack of a given operating system (OS). This feature gives possibility to assign the virtual honeypot with a so-called ‘personality’. The Personality Engine modifies a

packet's content in a way that it matches characteristics of the network stack of the previously configured OS. The adversary has the illusion that is receiving a response from the real OS.

Configuration Database

Configuration database is a module that provides the virtual honeypot with personality and required behavior. In the case when it is not specified for given IP address a default template is used. Configuration is assigned to the packet at the level of the Packet Dispatcher. The packet with specific configuration is then transferred to the Protocol Handler.

In the configuration file is included the personality of a given IP destination point which determines: the behavior of the network stack, the state of ports related to destination entity and protocol (open, block, reset); and the behavior of the specified port determined by the service script. There are a variety of commands used in configuration file as *create* which creates new template; the *set* which provides the personality from the file consisting fingerprints (nmap.prints) and *add* which changes the configuration of template. The *set* also defines the default behavior of the protocols. Additionally, the command *bind* is quite important because it finally grants the template an IP address in the network.

Routing Module

One of the most important features of honeyd is the capability to create wide range of virtual networks that seem to be real for adversary. Honeyd deceives the tools like traceroute and provides simulated topology. There is a possibility to even increase the reality of topology by adding fake link characteristic such as latency or packet loss. The Routing Module provides routing in simulated topology because there is not always possible to use Proxy ARP to direct the packets to the honeyd host. The routing is perceived as a tree with a root in an entry point to the simulated topology. Each node is treated like

a router and each edge as a link with modifiable characteristic. Terminal nodes of the tree correspond to networks. Honeyd supports multiple entries points to the topology. An entry router is chosen by the network space for which it is responsible. When honeyd receives a packet, it chooses an appropriate entry point of the routing tree and forwards the packet to the destination point. The predefined link characteristics are taken into consideration to determine the delay of packet delivery.

The Routing Module supports also TTL parameter decrementing its value after each hoop and discarding the packet whenever TTL reaches 0. This component, additionally, allows the splitting the routing topology by using GRE tunneling. This is carried out to load balance across several honeyd installations by delegating parts of the address space to different honeyd hosts.

5.3 Functionality

Honeyd being a low interaction honeypot proves to have all advantages associated with honeypots: It is easy to install, configure and deploy. Additionally, it owns minimal risk, as the emulated services control what attackers can and cannot do. [23] Honeyd itself has no production value so every traffic which reaches this software should be perceived as highly suspected. Work of the honeypot is based on monitoring an unused IP pool and capturing the connections. When a connection attempt is made to an IP address that has no system, that connection is forwarded to honeyd. The alert message is generated depending on action made by adversary. In the next step, honeyd interacts with the intruder giving an illusion of the reality of whole system or topology. Generally, "*honeyd simulates the networking stack of different operating systems and can provide arbitrary routing topologies and services for an arbitrary number of virtual systems.*" [22]

By default, honeyd detects and logs any connection to any UDP or TCP port. The possibilities of honeyd can be widen by the service scripts which are associated with specific ports. They are used to monitor traffic on specific port giving possibilities to further interaction with intruder. These scripts are written in

a way that they should know how to react for specific conditions. The main drawback of service scripts is that it is not always possible to predict every move of the adversary. It can happen that sometimes the intruder may act in an unrecognizable way for a given script, what usually causes an error message. In some cases, service scripts can be used to reveal the honeypot existence causing the cancelation of further interactions by the intruder. There are few already ready scripts available at the honeyd webpage but any user of honeyd can feel free to participate in honeyd development by writing own scripts meeting their particular demands.

5.4 ARP Deamon

Even though the honeyd is very powerful tool itself it needs a little help to provide all aforementioned functionality. Generally honeyd runs together with ARP daemon (for Debian GNU/Linux is named farpd), a program which replies to any ARP request for an IP address matching the specified destination with the hardware MAC address of the host on which the software is installed. This functionality works perfectly with honeyd. Every time, when someone tries to establish connection with an IP address, the ARP request is sent. In normal conditions the host owning this IP responds with its packets. Running farpd causes that if no one responds to the ARP request, the farpd does it, showing the host on which is installed as a owner of that IP. In this way all traffic to unallocated IP pool in the network will be processed by the honeypot. Any IP address claimed by farpd is eventually forgotten after a period of inactivity or after a hard timeout, and it is relinquished if the real owner shows up.

5.5 Configuration

Honeyd allows users to widely customize its configuration for meeting their own requirements. The configuration is done very simply by creating a file (Default.conf) which includes templates for a completely configured computer

system personalities, network topology, routing, IP addresses, ports, service scripts and behavior associated with each one of that elements.

5.6 GRE Tunneling

Honeyd gives possibility to create more advanced routing topology using GRE (Generic Routing Encapsulation), a tunneling protocol able to encapsulate network layer protocols inside virtual point-to-point links over the Internet Protocol. It can be used to load balance across several honeyd hosts granting them parts of the address space. Additionally, this functionality enables to delegate networks that belong to separate parts of the address space to a single honeyd host.

5.7 Fingerprints

Fingerprint element of honeyd functionality is very important solution to deceive tools like personality and port scanners - Nmap or Xprobe. These tools are used by hackers to gather information about target system in the network (mainly personality – the type of OS run on target computer, ports - its state or services)

Honeyd uses Nmap fingerprint database as its reference for a personality's TCP and UCP behavior. Xprobe fingerprint database is used as reference for a personality's ICMP behavior. [22] The fingerprints used by honeyd are in files nmap.prints and xprobe.prints respectively. The following code presents one of the fingerprints:

```
Fingerprint Microsoft Windows XP Home Edition
Class Microsoft | Windows | NT/2K/XP | general purpose
TSeq(Class=RI%gcd=<6%SI=<23C4E&>330%IPID=I%TS=U)
T1(DF=Y%W=F424%ACK=S++%Flags=AS%Ops=MNW)
T2(Resp=Y%DF=N%W=0%ACK=S%Flags=AR%Ops=)
T3(Resp=Y%DF=Y%W=F424%ACK=S++%Flags=AS%Ops=MNW)
T4(DF=N%W=0%ACK=O%Flags=AR%Ops=)
T5(DF=N%W=0%ACK=S++%Flags=AR%Ops=)
T6(DF=N%W=0%ACK=O%Flags=AR%Ops=)
T7(DF=N%W=0%ACK=S++%Flags=AR%Ops=)
PU(DF=N%TOS=0%IPLEN=38%RIPTL=148%RID=E%RIPCK=E%UCK=E%ULEN=134%DAT=E)
```

The text string after ‘Fingerprint’ token is a name of that personality. The tokens ‘T(1-7)’ provide results of tests which are made to determine the stack behavior. The first test is the most significant. It determines how the network stack of the remote operating system creates the initial sequence number (ISN) for TCP SYN segments and determines how IP identification numbers and TCP timestamps are generated.

The main parameters which are used to create a OS fingerprint are:

- Initial packet size (16 bits)
- Initial TTL (8 bits)
- Window size (16 bits)
- Max segment size (16 bits)
- Window scaling value (8 bits)
- "don't fragment" flag (1 bit)
- "sackOK" flag (1 bit)
- "nop" flag (1 bit) [36]

Nmap

Nmap (*Network mapper*), is a port scanner created by Gordon Lyon. Additionally, it provides the names of the to-be services for scanned ports. Nmap implements a wide range of techniques to test TCP ports including nonstandard approaches as response to specification of implementation of network stacks which potentially can avoid firewalls or IDSes. Nmap is a active scanner so it generates much traffic that can lead to revealing its presence. There are several methods of active scanning. For example, Nmap uses

scanning based on full connection establishment. The result is a list of ports that nmap was able to connect to, with the names of services available on them.

However, the most known usage of Nmap tool is for OS detection using fingerprints TCP/IP stack. Nmap sends several probing packets either TCP and UDP and deeply scrutinizes the responses. After a wide range of tests like probing ISN TCP, analyze the option of TCP protocol, probing IPID and control initial size of the window, nmap compares the results with its own database containing huge amount of fingerprints of OSes. Every fingerprint includes a textual description of the system, producer name, system name, generation and type of device. [22] [23] [27]

Xprobe

Honeyd additionally is designed to deceive a second OS scanner – Xprobe. Xprobe, that was created and it is maintained by Fyodor Yarochkin and Ofir Arkin, allows for the remote OS identification. It is an active OS fingerprinting tool based on ICMP packets. Operation of this software is based on sending several ICMP packets to the target host and analyzing the response. The tool automates the logic of OS fingerprinting methods called "X". Xprobe proves to have a few advantages over the Nmap tool Firstly, what is very important in some situations, xprobe is faster – it needs no more than 4 packets to reveal a remote OS. Secondly, it can detect whether host is up – pinging is no longer necessary. Additionally, it does not send any malformed datagrams and it is able to distinguish many variants of Microsoft operating systems.

5.8 TCP/IP - Experimental Architecture

At the first stages of this dissertation, a small topology consisting of RSU and two OBUs in TCP/IP conditions was created. The RSU and OBU personalities were created by modifying randomly chosen fingerprint names, therefore, **it can not be perceived like simulation of WAVE environment**. It had purpose only to check the functionality of honeyd software.

The honeyd software was installed under Linux UBUNTU 10.10 running in Virtual Box. The host system was Windows Professional XP SP2.

Both OS received IP address in local network (192.168.1.39 Linux and 192.168.1.38 Windows). From the point of view of router they are perceived like regular clients.

Using honeyd, a very simple topology (192.168.2.0/24 network) was created. Fake router (RSU 192.168.2.1) with two fake nodes (OBUs, 192.168.2.40 and 192.168.2.41). The whole experimental network looks as depicted in figure 25.

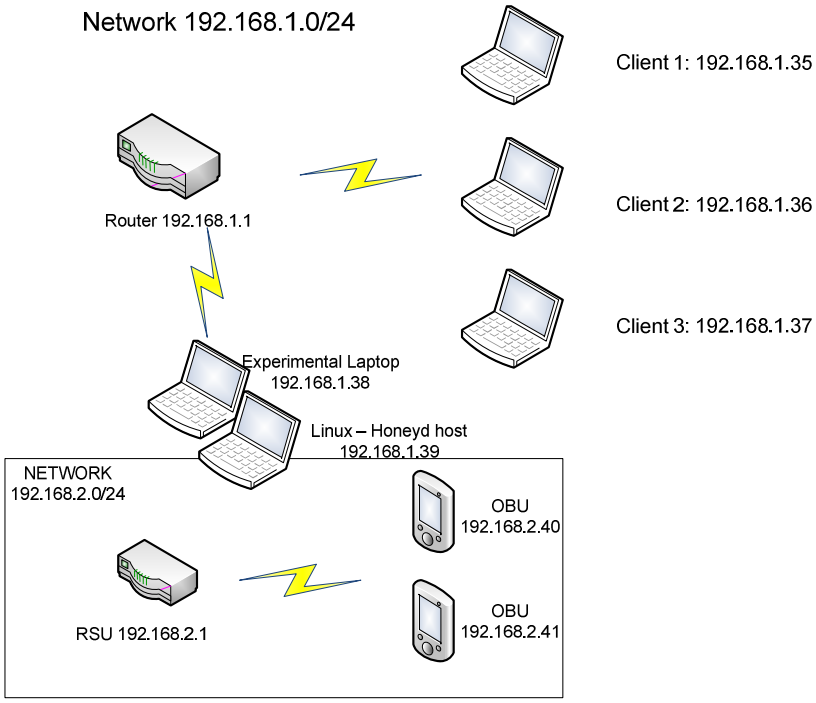

figure 25: Experimental Architecture

The configuration file of honeyd:

```
route entry 192.168.2.1 network 192.168.1.0/24
route 192.168.2.1 link 192.168.2.0/24

create OBU
set OBU personality "On Board Unit"
add OBU tcp port 80 "sh /usr/src/honeyd-1.5c/scripts/hello.sh"
add OBU tcp port 139 open
add OBU tcp port 137 open
```

```
add OBU udp port 139 open
add OBU udp port 137 open
set OBU default tcp action reset
set OBU default udp action reset

create RSU
set RSU personality "Road Side Unit"
set RSU default tcp action reset
set RSU default udp action reset
add RSU tcp port 23 "/usr/src/honeyd-1.5c/scripts/router-telnet.pl"
set RSU uid 32767 gid 32767
set RSU uptime 1327650

bind 192.168.2.40 OBU
bind 192.168.2.41 OBU
bind 192.168.2.1 RSU
```

The RSU is equipped with telnet-router.sh script, which provides router interface reachable from the console of any client in network (password and login)

The OBU is equipped with a giving the illusion of OBU interface (not yet clarified in details). The honeyd is run with ARP deamon, which responds to ping command for any unallocated IP.

In this configuration the honeyd is able to respond to commands like ping, telnet, nmap and xprobe. However, nmap and xprobe scanners do not own the names (personalities) ‘RSU’ and ‘OBU’ in theirs database so the match is not found in the case of scanning this topology.

Chapter 6

Vehicular Honeypot – Implementation of Honeyd

As previously stated honeypots open wide range of possibilities to improve security systems of vehicular environment. Implementation of this solution to this environment should own every feature of the standard honeypot: it should capture malicious data, be indistinguishable and able to data analysis. Vehicular honeypot can be implemented in VANET by proper modifications of honeyd.

6.1 Architecture of a Honeyd Software used as a Vehicular Honeypot

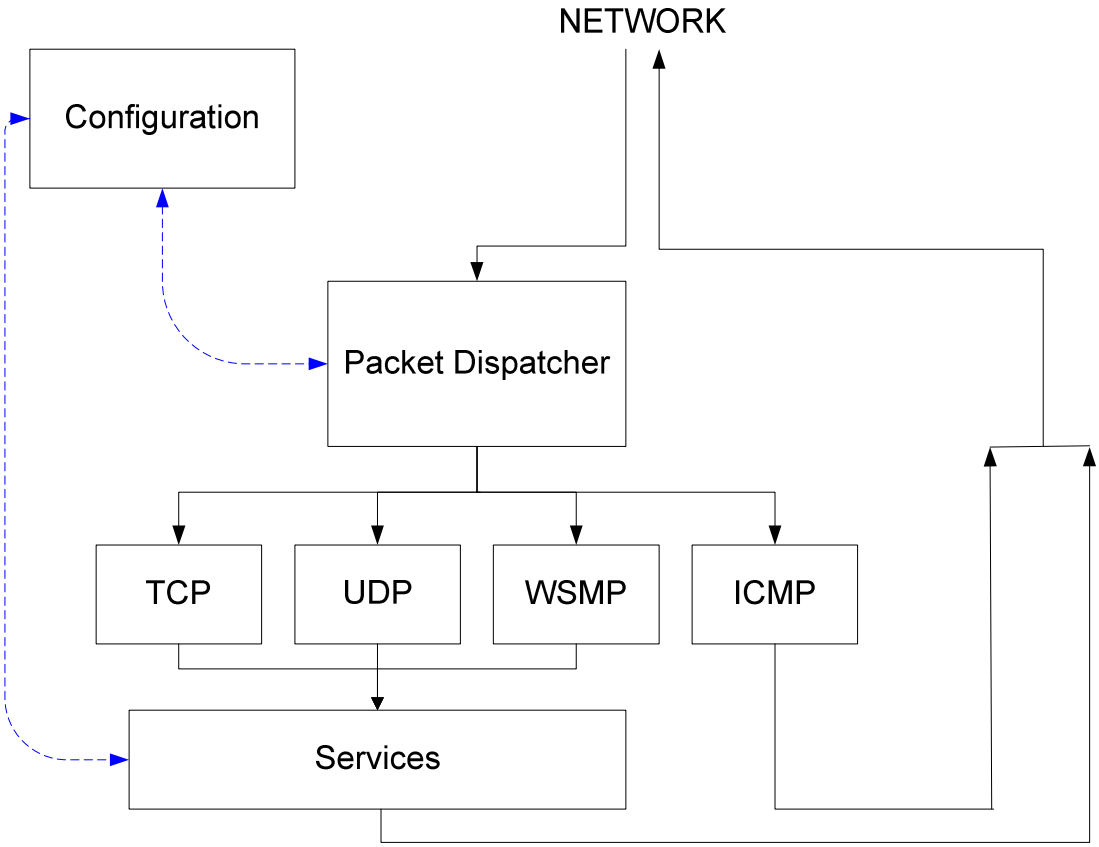

figure 26: Architecture of honeyd used as an OBU honeypot

As the functionality of honeyd used as vehicular honeypot differs from the original one, the architecture has to be modified to meet the requirements of WAVE. As figures 21 and 26 show, the main differences in architecture are noticeable in lacking of routing module in the case of vehicular honeypot.

The second principal difference in architecture is providing handling for the new WAVE oriented communication protocol - WSMP. It is very important module as communication in terms of WAVE mainly based on rapid and low overhead message exchange.

Additionally, the Personality Engine is also omitted what is explained in next sections. Other modules like the Packet Dispatcher remain but have to undergo deep modifications.

6.2 Functionality of Honeyd

Honeyd captures packets from USB interface by packet dispatcher. This module should recognize kind of protocol of incoming packet and associates it with configuration template which consists of application script being available on appropriate port.

In TCP/IP condition, the packet dispatcher retrieves from header of packets all important parameters like source and destination IP or port. Basing on these parameters, this module associates the incoming packet with appropriate host or script available on destination port. However, in the case of OBU (or RSU) honeypot, the configuration template is not associated with any IP address. The changeable link local IPv6 address is associated only with an OBU.

The packet is transmitted to honeyd where header of IPv6 is discarded (but logged and stored) whereas the header of UDP or TCP is analyzed. The honeyd should check the port number of the header of UDP packet as it is more likely used in WAVE. The number of port has to be known to reach the destination application. The fake packet produced by application script in cooperation with the Protocol Handler is transmitted to OBU where is encapsulated with IPv6 header and send to the destination via air-link

Concerning WSMP packets the PSID value should be check to verify the application that should process the packet.

The application script produces appropriate response and sends it to OBU which then send it via air-link to entity basing on its MAC address. Additionally, at the beginning, the application residing in honeyd should send to OBU a PSID and port number values needed to produce WSA message.

6.3 Main modifications In Source Code

I would like to emphasize at this point of my work, that this dissertation is developed to provide general overview on changes and modifications which should be carried out in aim to implement a honeypot into WAVE. Hence, the purpose of this dissertation is not to provide ready-to-use source code but to propose modifications that would be required to implement a honeypot into WAVE in the future. It has to be to said that modifications proposed would required additional changes in source code to be fully developed, however, this out of scope of this dissertation. It is justified by the fact that there is no working OBU device that can be used to interact with modified honeyd software to verify or check changes which would be done.

The source code is available at:
http://code.google.com/p/honeyd/source/browse/#svn%2Ftrunk%2Fhoneyd

Interface

Honeyd analyzes packets incoming from specified interface (wireless network card, Ethernet). In WAVE condition packets will be originated from physical interface of OBU (or RSU). Then, they will be transported via USB.

Hence, the honeyd software requires adding USB interface module to provide messages exchange between honeyd and OBU (or RSU) and then, external, network.

IPv4 and IPv6 headers

Honeyd is mainly designed to work with IPv4 packets. The software retrieves directly packets form one of the interfaces and process them. However, honeyd as an OBU (or RSU) honeypot does not retrieve incoming packets directly from network. The OBU device is responsible for that. The IP addressing is not necessary for honeyd to work properly. The only important parameter retrieved from incoming packets via USB is port number of destination and source application or PSID in the case of WSM. The binding personality with IP address as well as retrieving IP addresses from headers should be abandoned. However, the IPv6 header is very important in context of identifying source of packet so it should be logged and stored by honeyd.

Hence, the following changes have to be provided:

In honeyd.c file should be added a function to discarding IPv6 header of packet and sending it to logging module.

Packet Dispatcher

Honeyd uses tcpdump program to recognize incoming packets. This program allows dumping traffic on a network. It is able to examine IPv4, IPv6, UDP, TCP, ICMP. It can be used to print out the headers of packets on a network interface or filter packets that match a certain expressions.

Tcpdump uses the libpcap external linux library where pcap is a simplified object-oriented Python wrapper.

Honeyd uses a Pypcap as a Python interface to the pcap.

The one of the biggest challenges is to “tailor” honeyd software to fit requirements of WSMP, as WAVE supports two network layer protocols IPv6 and WSMP.

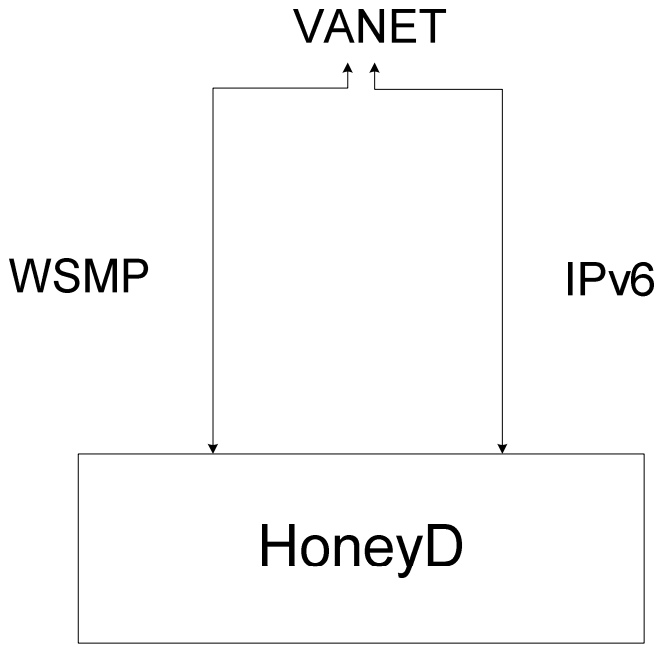

figure 27 Honeyd Communication Protocols

The processing of WSMP packet should look as follows:
From the incoming WSM, the PSID value is retrieved and the match with residing application is verified. If the match is found the WSM will be processed to application. Capturing module should be modified with a function that would recognize headers of WSMP packets. The function in honeyd.c file responsible for retrieving PSID value from WSM header should be provided.

Condition.c file has to be equipped with the function which will be responsible for associating a packet with application but not as it was in the case of UDP and TCP basing on IP addresses but basing on PSID of application/service or port number in the case of UDP and TCP.

Additionally, as pcap module should capture incoming packets via the USB it seems that changes are necessary to add USB interface module that pcap would be able to retrieve packets from it.

Hence, the following changes have to be provided:

The files pcap.c and pcap.h has to be modified to delete network interfaces and add USB interface as one and only way of receiving packets.

The following module can be implemented as a pcap – USB interface:
http://www.opensource.apple.com/source/libpcap/libpcap-23/libpcap/pcap-usb-linux.c

Protocol Handler

Outcoming packet should be processed and generated by the application in cooperation with the Protocol Handler. Honeyd will forward it via USB to OBU (or RSU) where it will be send to network.

Hence, the following changes have to be provided:

The creation of code for WSMP handling (similar to TCP.c or UDP.c) is necessary to provide processing of WSMP packets. The proposed name for such file is WSMP.c and WSMP.h with prototypes of functions.

Application

As honeyd gives opportunity to imitate some services by responding with fake packets to requests like ping or telnet. The vehicular honeypot has to be able to respond for request for service originated from network or intruder.

A specified, attention attracting, application/service has to be associated with the vehicular OBU (or RSU) honeypot, and the honeyd has to be modified with the creation/parsing file to provide fake responses.

Additionally the OBU (or RSU) honeypot has to be able to execute all management concerning application registration or WBSS management as stated in IEEE 1609.3 including broadcasting WAVE Service Advertisement (WSA).

Honeyd hosting application has to send to OBU`s (or RSU) management entity the PSID and port value necessary for WSA broadcasting and connection establishing.

A module responsible for creating responding packets is a dpkt. Dpkt is a fast, simple packet creation / parsing, with definitions for the basic, TCP/IP protocols and then some.

Hence, the following changes have to be provided:

There has to be written new code to provide processing WSMP. Proposed file is: wsmp.py.

Dkpt listing could be possible narrowed to just following files:

init.py

Dpkt.py

Udp.py

Tcp.py

Wsmp.py

Icmp6.py

Ip6.py

The most important functionality of packet creation for application is sending to OBU (or RSU) in aim to create WSA, the PSID and port number that application is available on. Creation the more complex packets being response to interaction with application is dependent on complexity of application residing by honeyd.

Personality Engine

As the behavior of honeypot`s network stack is determined by personality engine by providing changes into the protocol headers of outgoing packets to match the characteristic of the configured operating system, the personality engine for WAVE entities has to be consider.

The honeyd provides the TCP/IP stack fingerprints for wide range of OSs which is a collection of configuration attributes retrieved from a remote system during standard layer 4 network communications. However, in context of OBU honeypot the case of using honeyd differs widely. It should be remembered that honeyd is working with real OBU (or RSU) and is simulating only application at appropriate port. Hence, honeyd does not simulate network stack of any OS. Therefore, all fingerprinting and personalities are not valid because all packet would be transmitted from OBU (or RSU), not from interface of OS hosting honeyd.

What is more, even port scanning tools like nmap are useless as any application of provider would advertise itself within WSA providing all interested users with port numbers on which appropriate application is available.

Hence, the following changes have to be provided:

Files being responsible for providing personality engine and fingerprints for entities have to be deleted.

generate_assoc.py
nmap.assoc
nmap.prints
osfp.c

osfp.h
personality.c
personality.h
pf.c
pf_osfp.c
pfctl_osfp.c
xprobe2.conf
xprobe_assoc.h
xprobe_assoc.c

The changes in honeyd.c concerning aforementioned deletion should be done.

Behaviour of vehicle

To maintain the realism of node, and to provide a security for a driver separating real ECUs of the car from the OBU, the script producing fake messages has to be developed. These messages should concern events like sudden breaking, sharp bends or rain indication. These scripts have to pretend to be originated from real in-vehicle network (ECUs). However, writing these scripts is out of this work.

Routing Module

This module should be abandoned as the OBU (or RSU) honeypot will not be simulating topology.

Hence, the following changes have to be provided:

The file responsible for creating routing tree with configured parameters has to be deleted. It is router.c and router.h

The changes in honeyd.c concerning aforementioned deletion should be done.

GRE Tunneling

As well as a routing module the GRE tunneling functionality should also be removed as OBU (or RSU) honeypot will not be simulating routing.

Hence, the following changes have to be provided:

The file gre.c and gre.h (honeyd listing) should be removed.

The changes in honeyd.c concerning aforementioned deletion should be done.

ARP (Address Resolution Protocol)

In WAVE conditions AR protocol is replaced by Neighbor Cache protocol what is connected with usage of IPv6. IPv6 has provisions neighbor discovery in which a Neighbor Cache is populated with IP addresses and associated MAC addresses of devices within communications range. However, the device responsible for handling this protocol is OBU (or RSU), hence, this functionality of honeyd is no more valid.

Hence, the following changes have to be provided:

Following files can be deleted from software: arp.c and arp.h. (honeyd listing)

The changes in honeyd.c concerning aforementioned deletion should be done.

Configuration

The pattern of configuration of templates should be changed to simulate an application reached on any available port.

Hence, the following changes have to be provided:

File config.c have to be modified to meet requirements of simulating WAVE application. The possibilities of configuration file should be narrowed to binding services (application scripts) to respective ports.

DHCP client

Honeyd supports acquiring IP addresses via DHCP, however in terms of WAVE it is useless.

Hence, the following changes have to be provided:

The files dhcp.c and dhcp.h should be deleted.

The changes in main function concerning aforementioned deletion should be done.

Ethernet

Ethernet is also no supported.

Hence, the following changes have to be provided:

Files ethernet.c and ethernet.h (honeyd lisiting)can be removed from software.

The changes in main function concerning aforementioned deletion should be done.

Chapter 7

Conclusion and Future Work

The aim of this dissertation was designing honeypot device in WAVE environment. Vehicular honeypots, either OBU or RSU should own following features:

- Have to be indistinguishable from other nodes and protected from exposure and from physical stealing
- Have to be able to interact with intruder (or normal users) or generate and simulate random traffic (messages) in way that external observer will not be able to found out that the traffic is fake.
- Be able to analyze the incoming traffic, assuming that the incoming packets are not only malicious attempts.
- Should own full functionality as regular entity

In addition, they should play following roles:

- Simulate nodes of network (OBU, RSU)
- Catch all malicious attempts against its resources
- Analyze malicious attempts in aim to improve security system

Proposed designs: laptop with honeypot software connected to OBU device (in the case of OBU honeypot) or to RSU device (in the case of RSU honeypot) are enough to satisfy such requirements. The biggest challenge was to design proper honeypot software meeting requirements of highly specific vehicular environment. Honeypot software should imitate application layer of WAVE stack and be able to process communication protocols like WSMP and UDP. It should communicate with OBU (or RSU) via USB to provide interaction with network. Above all, it should log and store every packet received or sent. OBU honeypot equipped with such software would drive around some area catching all malicious attempts to break into its resources. The solution was based on already existing software working in TCP/IP environment: honeyd. This particular software was chosen because it proved to have every functionality of good honeypot and additionally was elastic enough to be implemented in WAVE. However, as it can be assumed from proposed designs and architectures of vehicular honeypot, functionality of honeyd in TCP/IP and WAVE differ widely. The reach and tremendously useful functionality of honeyd is narrowed in WAVE to partially simulating application layer. Hence, the source code was deeply analyzed to determine modules which have to undergo modifications in aim to meet requirements of WAVE. As a result, modules like: routing module or personal engine, used in TCP/IP local networks were abandoned due to nature of WAVE implementation.

However, the advantage of honeyd can be still successfully used in WAVE provided that the modifications in other modules will be done. The most important are:

- Creating WSMP and IPv6 protocols handling
- Creating USB interface
- Improving Packet Dispatcher to recognize WAVE packets
- Improving service scripts to become WAVE application

The designing of application script residing within honeyd is a future work and it depends of WAVE implementation. The functionality of such application can vary widely depending of local requirements. Additionally, the application available within OBU and RSU would be of different kind.

Next future work which has to be considered is designing software simulating ECUs and its real behavior. Cooperation of modified honeyd with such software would provide full and complex implementation of vehicular honeypot.

Bibliography

1. P. Verissimo, L. Rodrigues, “Distributed Systems For System Architects”, Kluwer Academic Publishers, 2001
2. I. Koren, C.M. Krishna, “Fault Tolerant Systems”, Elsevier, Inc, 2007, Chapter 9
3. Ch. Paar, J Pelzl, “Understanding Cryptography“ , Springer, 2010,
4. IEEE Vehicular Technology Society, “IEEE Trial-Use Standard for Wireless Access in Vehicular Environments (WAVE)—Resource Manager”, Std 1609.1™, USA, 2006
5. Intelligent Transportation Systems Committee, “IEEE Trial-Use Standard for Wireless Access in Vehicular Environments—Security Services for Applications and Management Messages”, Std 1609.2™, USA, 2006
6. Intelligent Transportation Systems Committee, “IEEE Trial-Use Standard for Wireless Access in Vehicular Environments (WAVE)—Networking Services”, Std 1609.3™, USA, 2007
7. IEEE Vehicular Technology Society, “IEEE Trial-Use Standard for Wireless Access in Vehicular Environments (WAVE)—Multi-channel Operation“, Std 1609.4™, USA, 2006
8. IEEE Computer Society, “IEEE Standard for Information technology—Telecommunications and information exchange between systems—Local and metropolitan area networks— Specific requirements. Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications”, Std 802.11, USA, 2007
9. IEEE Computer Society, “IEEE Standard for Information technology—Telecommunications and information exchange between systems—Local and metropolitan area networks— Specific requirements. Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications. Amendment 6: Wireless Access In Vehicular Environment”, Std 802.11p, USA, 2010
10. T. Weil, “Securing Wireless Access in Vehicular Environments (WAVE)”, Booz Allen Hamilton, New Orleans, December 1th, 2008

11. P. Papadimitratos, L. Buttyan, T. Holczer, E. Schoch, J. Freudiger, M. Raya, Z. Ma, F. Kargl, A. Kung, J.-P. Hubaux, “Secure Vehicular Communication Systems: Design and Architecture”, IEEE Communications Magazine, November 2008
12. M. Wolf, “Vehicular Security Hardware: The Security for Vehicular Security Mechanisms”, Embedded Security in Cars Conference (*escar*), Hamburg, November 18th, 2009
13. F. Kargl, P. Papadimitratos, L. Buttyany, M. Muter, E. Schoch, B. Wiedersheim, T.V. Thongy, G. Calandriello, A. Heldz, A. Kung J.-P. Hubaux, “Secure Vehicular Communication: Systems: Implementation, Performance, and Research Challenges”, Communications Magazine, IEEE, November 2008
14. Raúl Siles “HoneySpot: The Wireless Honeypot”, The Spanish Honeynet Project (SHP), December 17, 2007
15. Laurent Oudot, “Wireless Honeypot Countermeasures”, http://www.symantec.com/, 2010
16. Lance Spitzner, “Honeypots, Definitions and Value of Honeypots”, http://www.tracking-hackers.com, 2003
17. V. Verendel, D. K. Nilsson, U. E. Larson, and E. Jonsson, ”An Approach to using Honeypots in In-Vehicle Networks”, 2008
18. Karl Koscher, Alexei Czeskis, Franziska Roesner, Shwetak Patel, and Tadayoshi Kohno, “Experimental Security Analysis of a Modern Automobile”, IEEE Symposium on Security and Privacy, 2010
19. Daniel Jiang, Luca Delgrossi, “IEEE 802.11p: Towards an International Standard for Wireless Access in Vehicular Environments”, Vehicular Technology Conference, 2008, May 2008
20. Federal Highway Administration, “VII Architecture and Functional Requirements”, July 2005
21. Maxim Raya, Jean-Pierre Hubaux, “Securing vehicular ad hoc networks“, Journal of Computer Security - Special Issue on Security of Ad-hoc and Sensor Networks, January 2007
22. Niels Provos, “A Virtual Honeypot Framework”, October 21, 2003
23. Lance Spitzner, “Honeypots: Tracking Hackers”, chapter 8, September 20th, 2002

24. Yunxin (Jeff) Li, “An Overview of the DSRC/WAVE Technology”, 7th International ICST Conference on Heterogeneous Networking for Quality, Reliability, Security and Robustness, 2010
25. Dr. Michele Weigle, ”Standards: WAVE / DSRC / 802.11p”, Old Dominion University, 2008
26. Krzysztof Modzelewski, “Czym jest ITS”, accessed in 2011/01/02 http://www.itspolska.pl/?page=11
27. Gordon Lyon, http://www.nmap.org, accessed in 2011/05/20
28. “Intelligent_transportation_system”, http://en.wikipedia.org/wiki/Intelligent_transportation_system, accessed in 2011/03/15
29. “Vehicular ad-hoc network”, http://en.wikipedia.org/wiki/Vehicular_ad-hoc_network, accessed in 2010/11/09
30. Institute of Electrical and Electronics Engineers, “IEEE 1609 - Family of Standards for Wireless Access in Vehicular Environment (WAVE)”, http://www.standards.its.dot.gov/fact_sheet.asp?f=80, accessed in 2011/02/21
31. “Applications Overview”, http://www.itsoverview.its.dot.gov, accessed in 2011/05/20
32. “DSRC & WAVE”, http://www.slidefinder.net/c/ch3_20dsrc/19131537 accessed in 2011/02/07
33. “Elliptic curve cryptography”, http://en.wikipedia.org/wiki/Elliptic_curve_cryptography, accessed in 2011/01/10
34. “Symmetric Key Cryptography”, http://library.thinkquest.org/07aug/01676/relevance_cryptographictechnologies_encryption_symmetric.html, accessed in 2010/11/29
35. “fakeAP”, http://www.blackalchemy.to/project/fakeap, accessed in 2011/04/10
36. “IP stack fingerprinting” http://en.wikipedia.org/wiki/TCP/IP_stack_fingerprinting, accessed in 2010/05/10

37. Jonathan Burr, “Elliptical curve cryptography (ECC)”, http://searchsecurity.techtarget.com/definition/elliptical-curve-cryptography, accessed in 2011/08/15